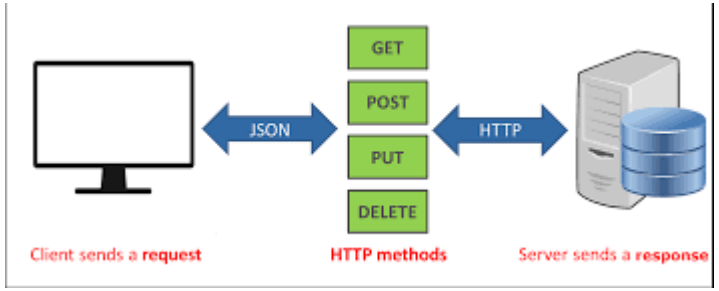
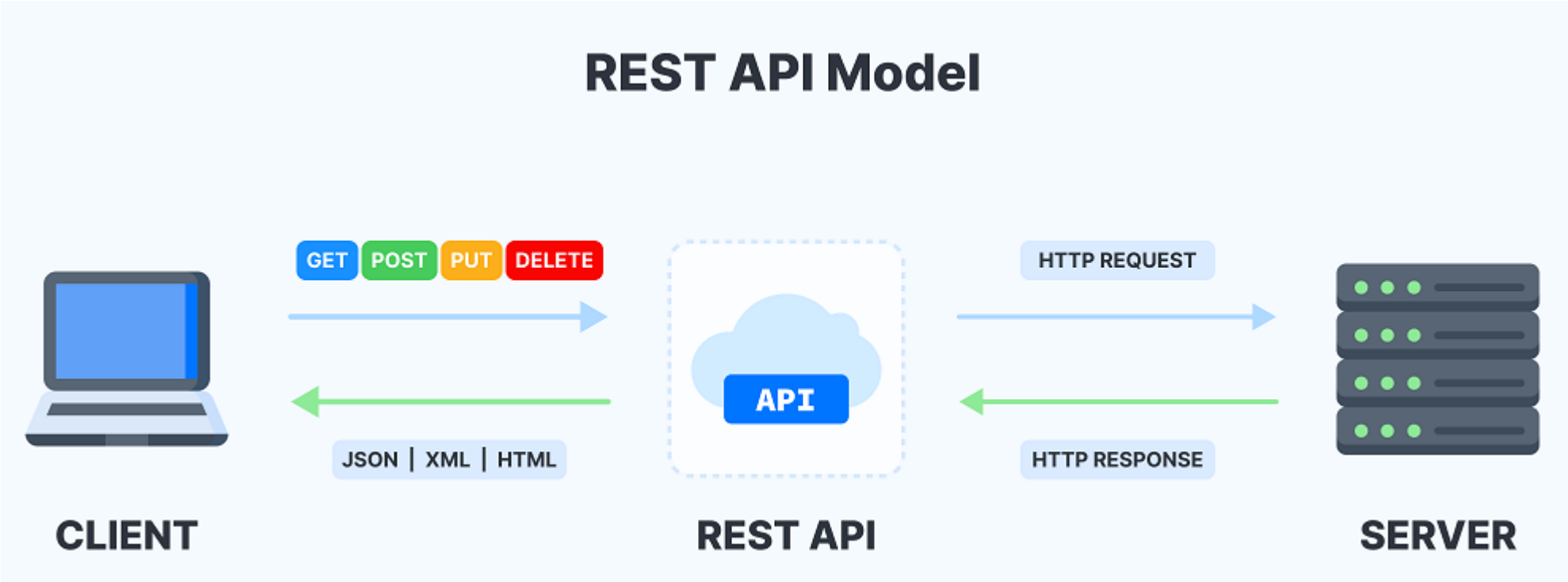
REST API


GET
POST
PUT
DELETE
JSON
HTTP
Client sends a request
HTTP methods
Server sends a response


REST API Model
GET
POST
PUT
DELETE
JSON | XML | HTML
API
HTTP REQUEST
HTTP RESPONSE
CLIENT
REST API
SERVER

Qué es REST – API ?

Para que una API se considere de RESTful, debe cumplir los siguientes criterios:

• Arquitectura cliente-servidor con solicitudes a través de HTTP.

• Comunicación entre el cliente y el servidor sin estado, las solicitudes son independientes.

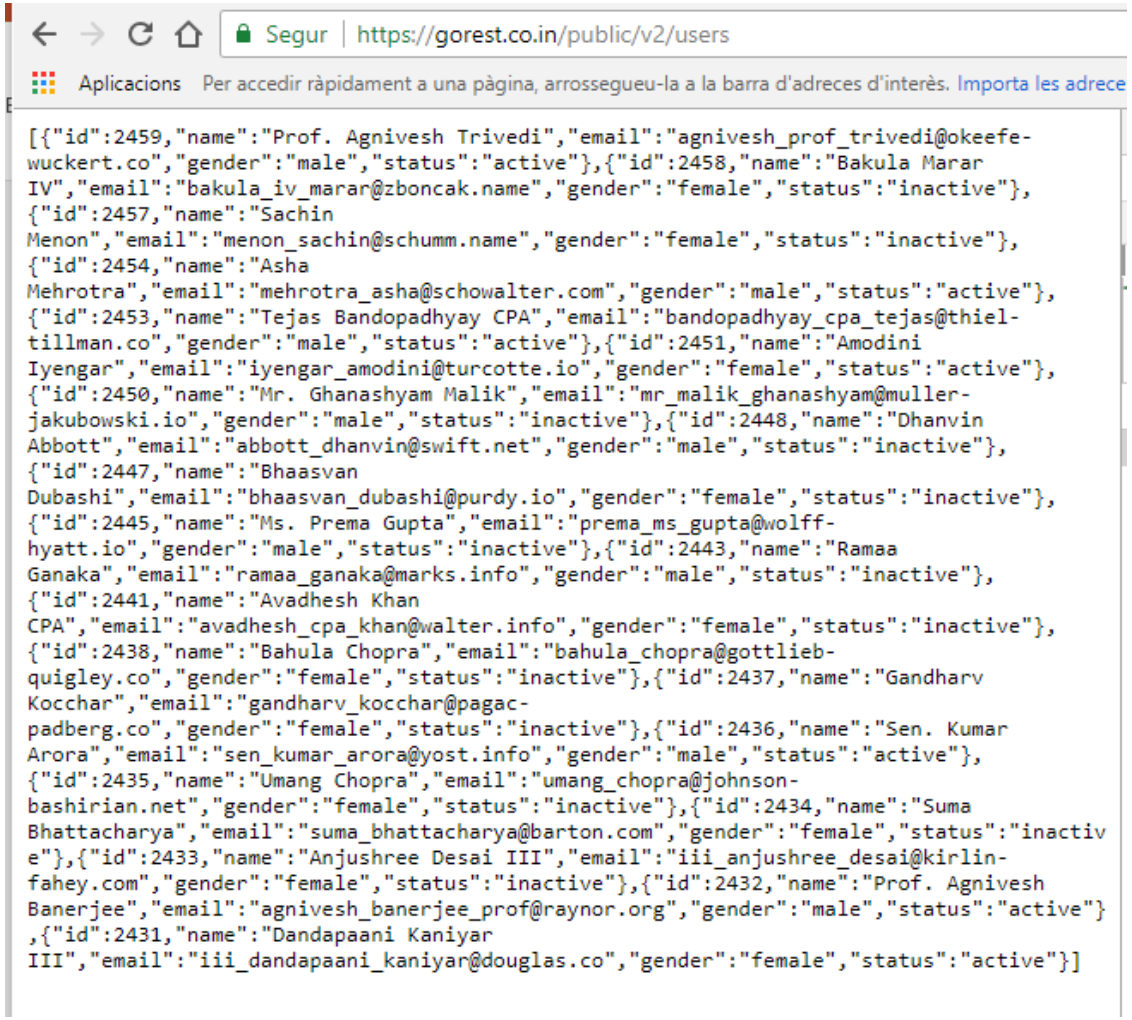
• Una interfaz uniforme entre los elementos, para que la información se transfiera de forma estandarizada.

• Para ello deben cumplirse las siguientes condiciones:

- Los recursos solicitados deben ser identificables e independientes de las representaciones enviadas al cliente.
- El cliente debe poder manipular los recursos a través de la representación que recibe, ya que esta contiene suficiente información para permitirlo.
- Debe contener hipertexto o hipermedios, lo cual significa que cuando el cliente acceda a algún recurso, debe poder utilizar hipervínculos para buscar las demás acciones que se encuentren disponibles en ese momento.

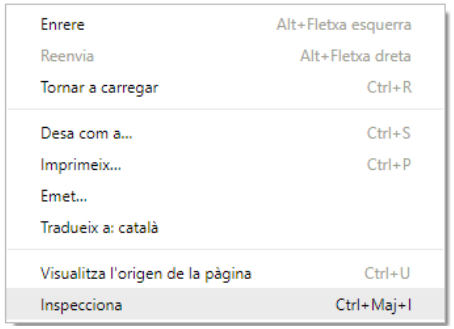
Usando el navegador Chrome para hacer llamadas y ver el resultado

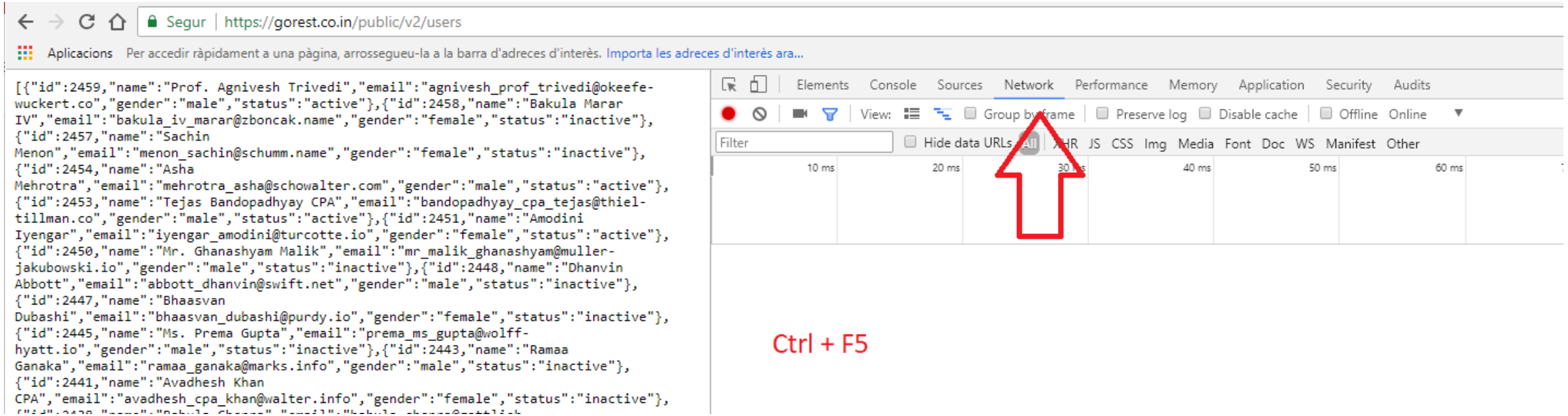
https://gorest.co.in/public/v2/users



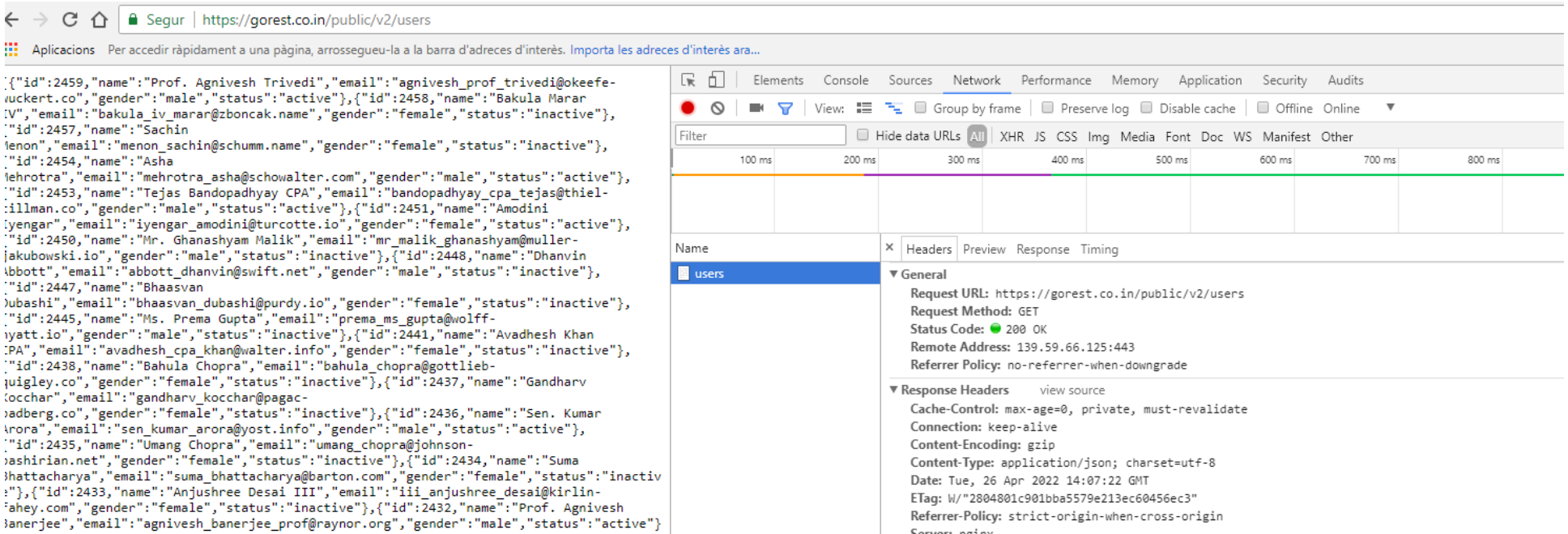
[{"id":2459,"name":"Prof. Agnivesh Trivedi","email":"agnivesh_prof_trivedi@okeefe-wuckert.co","gender":"male","status":"active"},{"id":2458,"name":"Bakula Marar IV","email":"bakula_iv_marar@zboncak.name","gender":"female","status":"inactive"},{"id":2457,"name":"Sachin Menon","email":"menon_sachin@schumm.name","gender":"female","status":"inactive"},{"id":2454,"name":"Asha Mehrotra","email":"mehrotra_asha@schowalter.com","gender":"male","status":"active"},{"id":2453,"name":"Tejas Bandopadhyay CPA","email":"bandopadhyay_cpa_tejas@thiel-tillman.co","gender":"male","status":"active"},{"id":2451,"name":"Amodini Iyengar","email":"iyengar_amodini@turcotte.io","gender":"female","status":"active"},{"id":2450,"name":"Mr. Ghanashyam Malik","email":"mr_malik_ghanashyam@muller-jakubowski.io","gender":"male","status":"inactive"},{"id":2448,"name":"Dhanvin Abbott","email":"abbott_dhanvin@swift.net","gender":"male","status":"inactive"},{"id":2447,"name":"Bhaasvan Dubashi","email":"bhaasvan_dubashi@purdy.io","gender":"female","status":"inactive"},{"id":2445,"name":"Ms. Prema Gupta","email":"prema_ms_gupta@wolff-hyatt.io","gender":"male","status":"inactive"},{"id":2443,"name":"Ramaa Ganaka","email":"ramaa_ganaka@marks.info","gender":"male","status":"inactive"},{"id":2441,"name":"Avadhesh Khan CPA","email":"avadhesh_cpa_khan@walter.info","gender":"female","status":"inactive"},{"id":2438,"name":"Bahula Chopra","email":"bahula_chopra@gottlieb-quigley.co","gender":"female","status":"inactive"},{"id":2437,"name":"Gandharv Kocchar","email":"gandharv_kocchar@pagac-padberg.co","gender":"female","status":"inactive"},{"id":2436,"name":"Sen. Kumar Arora","email":"sen_kumar_arora@yost.info","gender":"male","status":"active"},{"id":2435,"name":"Umang Chopra","email":"umang_chopra@johnson-bashirian.net","gender":"female","status":"inactive"},{"id":2434,"name":"Suma Bhattacharya","email":"suma_bhattacharya@barton.com","gender":"female","status":"inactive"},{"id":2433,"name":"Anjushree Desai III","email":"iii_anjushree_desai@kirlin-fahey.com","gender":"female","status":"inactive"},{"id":2432,"name":"Prof. Agnivesh Banerjee","email":"agnivesh_banerjee_prof@raynor.org","gender":"male","status":"active"},{"id":2431,"name":"Dandapaani Kaniyar III","email":"iii_dandapaani_kaniyar@douglas.co","gender":"female","status":"active"}]

Botón derecho sobre un espacio en blanco de la pàgina, en Chrome Ctrl Maj I





Escoger pestanya Network y polsar Ctrl + F5 para que recargue la pàgina.



× Headers Preview Response Timing

▼ General

Request URL: https://gorest.co.in/public/v2/users
Request Method: GET
Status Code: 200 OK
Remote Address: 139.59.66.125:443
Referrer Policy: no-referrer-when-downgrade

▼ Response Headers view source

Cache-Control: max-age=0, private, must-revalidate
Connection: keep-alive
Content-Encoding: gzip
Content-Type: application/json; charset=utf-8
Date: Tue, 26 Apr 2022 14:07:22 GMT
ETag: W/"2804801c901bba5579e213ec60456ec3"
Referrer-Policy: strict-origin-when-cross-origin
Server: nginx
Strict-Transport-Security: max-age=63072000; includeSubDomains
Transfer-Encoding: chunked
Vary: Origin
Vary: Accept-Encoding
X-Content-Type-Options: nosniff
X-Download-Options: noopen
X-Frame-Options: SAMEORIGIN
X-Links-Current: https://gorest.co.in/public/v2/users?page=1
X-Links-Next: https://gorest.co.in/public/v2/users?page=2

Filter | Hide data URLs | All | XHR | JS | CSS | Img | Media | Font | Doc | WS | Manifest | Other

100 ms | 200 ms | 300 ms | 400 ms | 500 ms | 600 ms | 700 ms | 800 ms | 900 ms | 1000 ms

Name | Headers | Preview | Response | Timing

users


```
▼[{id: 2459, name: "Prof. Agnivesh Trivedi", email: "agnivesh_prof_trivedi@okeefe-wuckert.co",…},…]
  ▼0: {id: 2459, name: "Prof. Agnivesh Trivedi", email: "agnivesh_prof_trivedi@okeefe-wuckert.co",…}
     email: "agnivesh_prof_trivedi@okeefe-wuckert.co"
     gender: "male"
     id: 2459
     name: "Prof. Agnivesh Trivedi"
     status: "active"
  ▶1: {id: 2458, name: "Bakula Marar IV", email: "bakula_iv_marar@zboncak.name", gender: "female",…}
  ▶2: {id: 2457, name: "Sachin Menon", email: "menon_sachin@schumm.name", gender: "female",…}
  ▶3: {id: 2454, name: "Asha Mehrotra", email: "mehrotra_asha@schowalter.com", gender: "male",…}
  ▶4: {id: 2453, name: "Tejas Bandopadhyay CPA", email: "bandopadhyay_cpa_tejas@thiel-tillman.co",…}
  ▶5: {id: 2451, name: "Amodini Iyengar", email: "iyengar_amodini@turcotte.io", gender: "female",…}
  ▶6: {id: 2450, name: "Mr. Ghanashyam Malik", email: "mr_malik_ghanashyam@muller-jakubowski.io",…}
  ▶7: {id: 2448, name: "Dhanvin Abbott", email: "abbott_dhanvin@swift.net", gender: "male",…}
  ▶8: {id: 2447, name: "Bhaasvan Dubashi", email: "bhaasvan_dubashi@purdy.io", gender: "female",…}
  ▶9: {id: 2445, name: "Ms. Prema Gupta", email: "prema_ms_gupta@wolff-hyatt.io", gender: "male",…}
  ▶10: {id: 2441, name: "Avadhesh Khan CPA", email: "avadhesh_cpa_khan@walter.info", gender: "female",…}
  ▶11: {id: 2438, name: "Bahula Chopra", email: "bahula_chopra@gottlieb-quigley.co", gender: "female",…}
  ▶12: {id: 2437, name: "Gandharv Kocchar", email: "gandharv_kocchar@pagac-padberg.co", gender: "female",…}
  ▶13: {id: 2436, name: "Sen. Kumar Arora", email: "sen_kumar_arora@yost.info", gender: "male",…}
  ▶14: {id: 2435, name: "Umang Chopra", email: "umang_chopra@johnson-bashirian.net", gender: "female",…}
  ▶15: {id: 2434, name: "Suma Bhattacharya", email: "suma_bhattacharya@barton.com", gender: "female",…}
  ▶16: {id: 2433, name: "Anjushree Desai III", email: "iii_anjushree_desai@kirlin-fahey.com",…}
  ▶17: {id: 2432, name: "Prof. Agnivesh Banerjee", email: "agnivesh_banerjee_prof@raynor.org", gender: "male",…}
  ▶18: {id: 2431, name: "Dandapaani Kaniyar III", email: "iii_dandapaani_kaniyar@douglas.co",…}
  ▶19: {id: 2430, name: "Ms. Dhanalakshmi Shah", email: "shah_dhanalakshmi_ms@skiles.co", gender: "male",…}
```

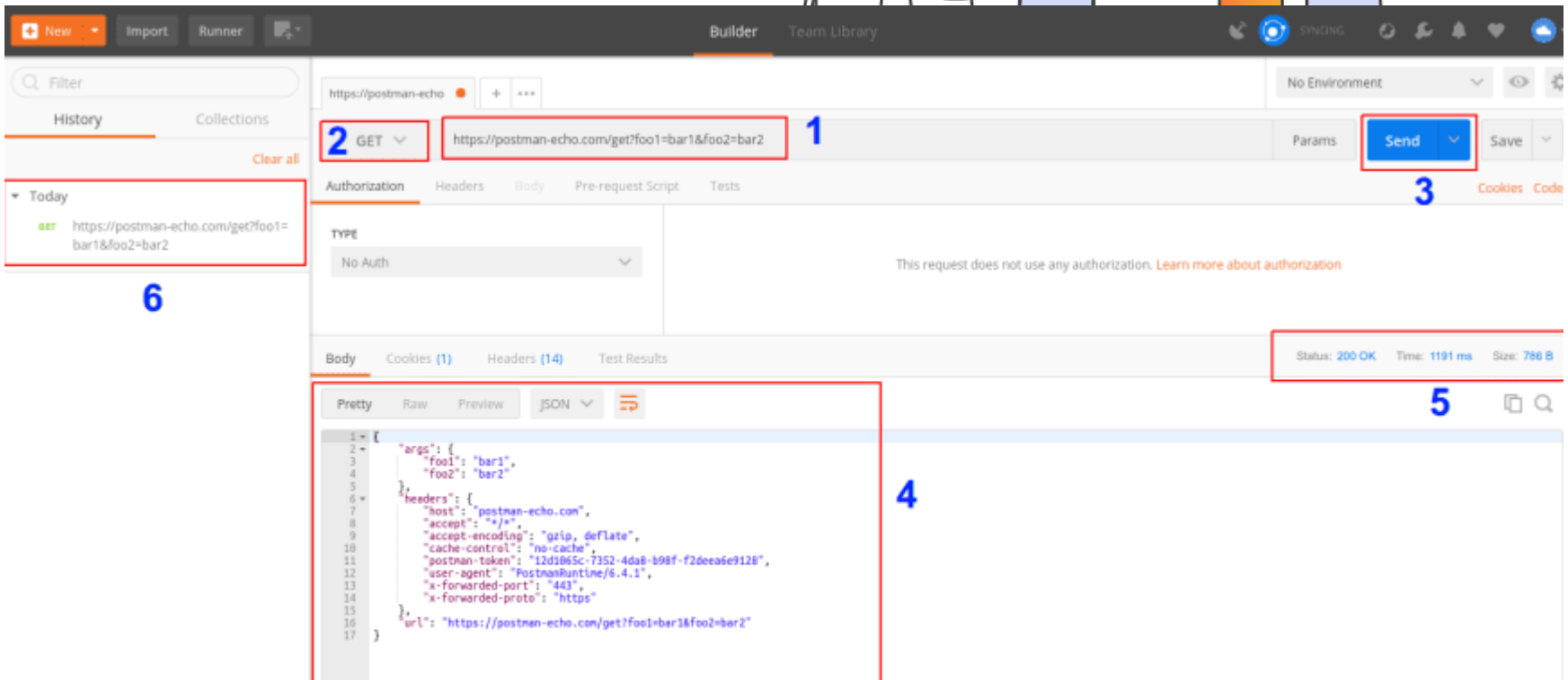
Aplicaciones para hacer pruebas REST sin programar

https://www.postman.com/product/rest-client/

Postman REST Client

Send requests, inspect responses, and easily debug REST APIs.

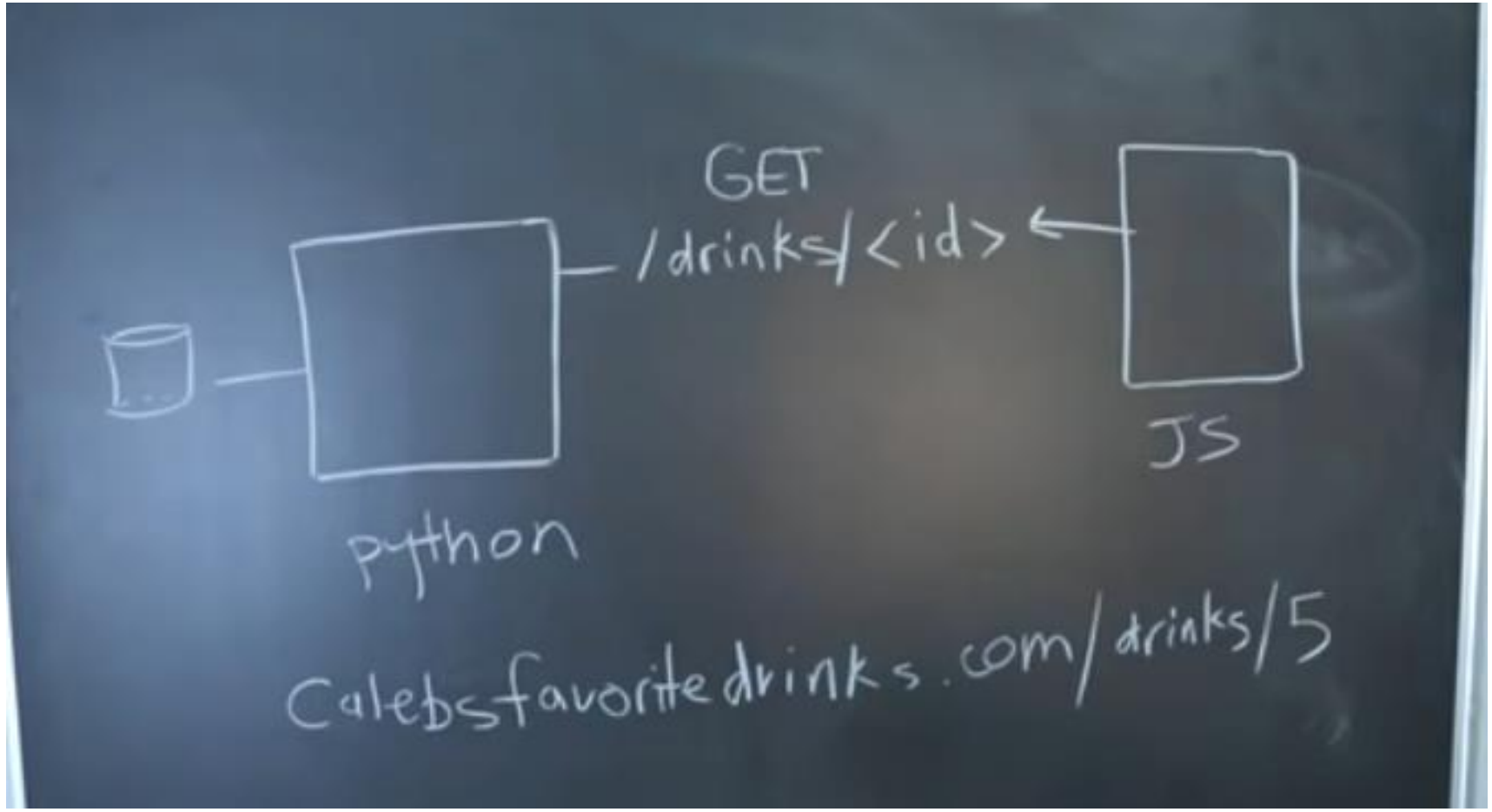
Download the App

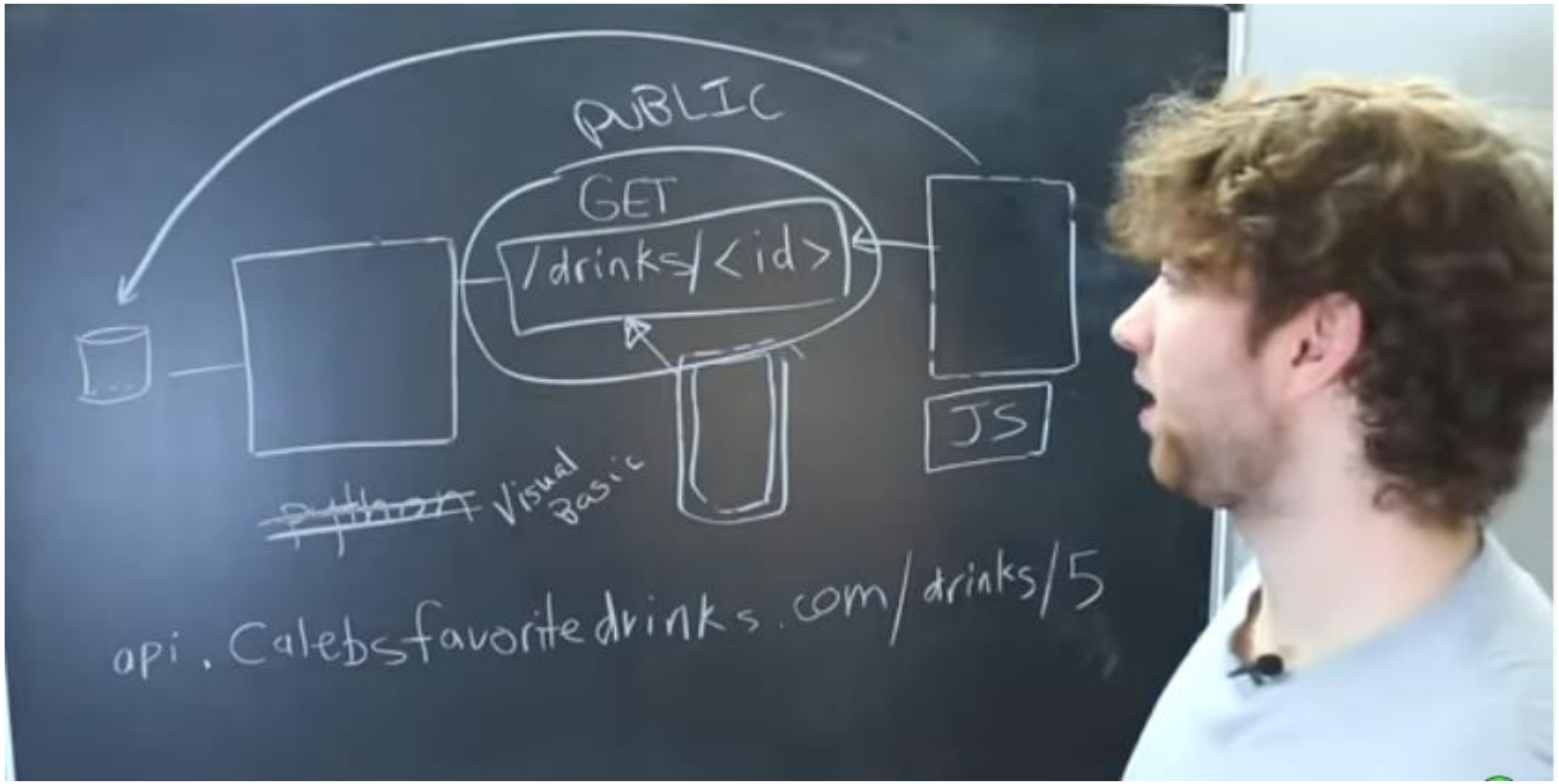


Se pueden programar llamadas en tiempo diferido y guardar los resultados en un archivo.

REST – API con Python

https://www.youtube.com/watch?v=qbLc5a9jdXo


GET
/drinks/<id>
JS
python
Calebsfavoritedrinks.com/drinks/5



Seguridad

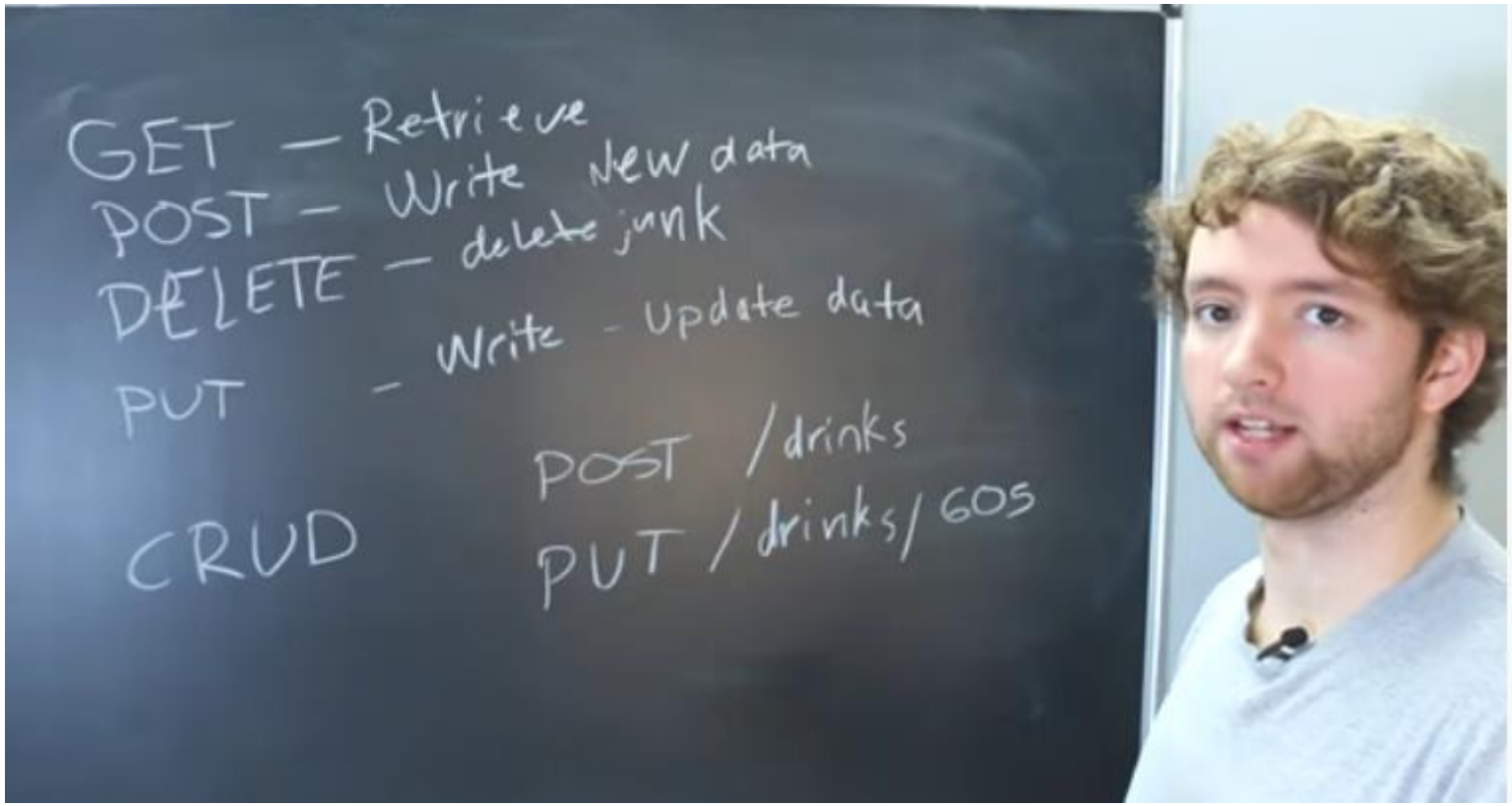
Modularidad

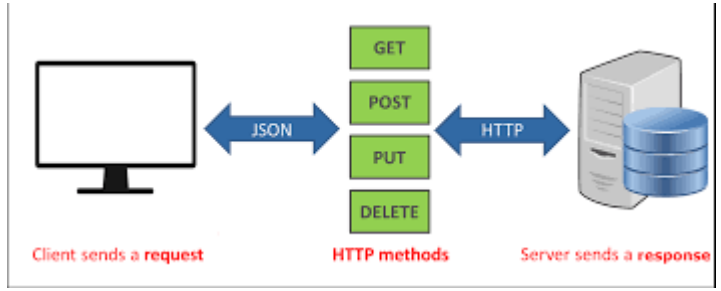
Aislamiento de cambios en server

Creación de nuevas interfaces ususario

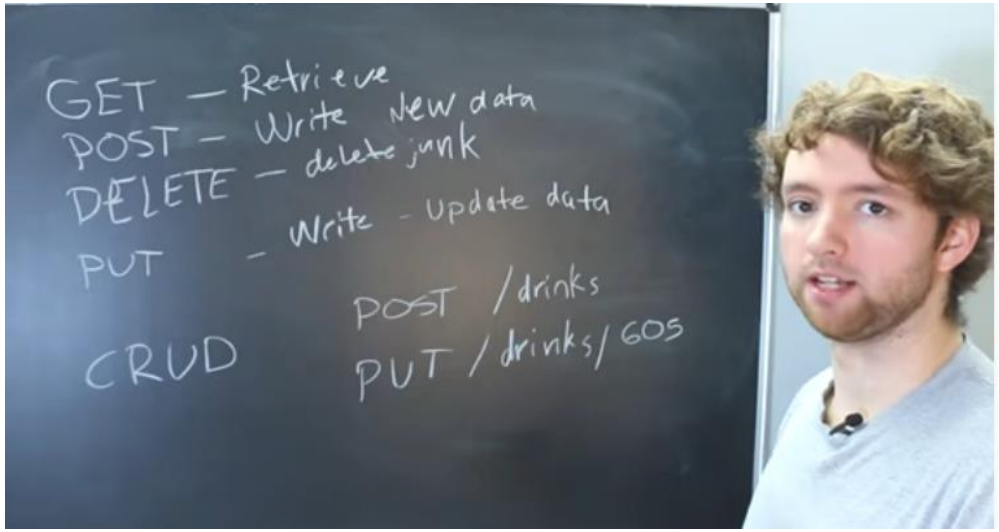
Automatización de procesos

https://www.youtube.com/watch?v=qbLc5a9jdXo


GET — Retrieve
POST — Write New data
DELETE — delete junk
PUT — Write - Update data
CRUD
POST /drinks
PUT /drinks/605


GET
POST
PUT
DELETE
JSON
HTTP
Client sends a request
HTTP methods
Server sends a response

Info:



Nota el video continua a partir del minuto 30 :

Creando un entorno virtual
Instalando flask
Y creando una base de datos de bebidas
Con flask crea las respuestas para mostrar la lista de bebidas y para ver una en concreto

```python
import requests
import json

response = requests.get('https://gorest.co.in/public/v2/users')
print(response.json())
```

```
[{'id': 3183, 'name': 'Gautam Trivedi', 'email': 'trivedi_gautam@considine-balistreri.com', 'gender': 'male', 'status': 'inacti
ve'}, {'id': 3181, 'name': 'Brahmabrata Adiga', 'email': 'adiga_brahmabrata@kutch.name', 'gender': 'female', 'status': 'activ
e'}, {'id': 3180, 'name': 'Deeptimayee Mehrotra', 'email': 'deeptimayee_mehrotra@quitzon.net', 'gender': 'female', 'status': 'i
nactive'}, {'id': 3179, 'name': 'Achyut Ahluwalia', 'email': 'achyut_ahluwalia@huel-ledner.net', 'gender': 'female', 'status':
'active'}, {'id': 3178, 'name': 'The Hon. Trilok Shah', 'email': 'trilok_hon_the_shah@dubuque.net', 'gender': 'male', 'status':
'inactive'}, {'id': 3177, 'name': 'Narayan Singh', 'email': 'narayan_singh@jast-swaniawski.co', 'gender': 'male', 'status': 'in
```

```python
import requests
import json

response = requests.get('https://gorest.co.in/public/v2/users')
#print(response)

for data in response.json():
    print(data)


```

```
{'id': 3183, 'name': 'Gautam Trivedi', 'email': 'trivedi_gautam@considine-bali
e'}
{'id': 3181, 'name': 'Brahmabrata Adiga', 'email': 'adiga_brahmabrata@kutch.na
{'id': 3180, 'name': 'Deeptimayee Mehrotra', 'email': 'deeptimayee_mehrotra@qu
ve'}
{'id': 3179, 'name': 'Achyut Ahluwalia', 'email': 'achyut_ahluwalia@huel-ledne
```

```python
import requests
import json

response = requests.get('https://gorest.co.in/public/v2/users')
# print(response.json())

for data in response.json():
    print(data)
```

```python
import requests
import json

response = requests.get('https://gorest.co.in/public/v2/users')
#print(response)

for data in response.json():
    print(data['id'], data['name'], data['email'])


```

```
3183 Gautam Trivedi trivedi_gautam@considine-balistreri.com
3181 Brahmabrata Adiga adiga_brahmabrata@kutch.name
3180 Deeptimayee Mehrotra deeptimayee_mehrotra@quitzon.net
3179 Achyut Ahluwalia achyut_ahluwalia@huel-ledner.net
3178 The Hon. Trilok Shah trilok_hon_the_shah@dubuque.net
3177 Narayan Singh narayan_singh@jast-swaniawski.co
3176 Divakar Verma Ret. ret_verma_divakar@brown.net
3175 Dinkar Mukhopadhyay Ret. dinkar_mukhopadhyay_ret@hammes.name
3174 Dinesh Dwivedi dwivedi_dinesh@yost-schmeler.net
3172 Ojaswini Dwivedi dwivedi_ojaswini@murazik.co
```

```python
import requests
import json

response = requests.get('https://gorest.co.in/public/v2/users')
#print(response)

for data in response.json():
    if data['status'] == 'active':
        print(data['id'], data['name'], '\t\t', data['status'])
```

```
3181 Brahmabrata Adiga          active
3179 Achyut Ahluwalia           active
3175 Dinkar Mukhopadhyay Ret.           active
3174 Dinesh Dwivedi             active
3172 Ojaswini Dwivedi           active
3169 Meena Nambeesan            active
3168 Devadatt Pandey III                active
3166 Deeptimayee Ahluwalia              active
3163 Bhadra Malik CPA           active
3162 Bhooshan Guha              active
```

PRACTICA P01. Leer json

https://jsonplaceholder.typicode.com/todos

1. Examina esta url.
2. Escribe un programa Python que lea todos los registros de la url y muestre los campos **`id`** y **`title`** de cada registro.

```
1 delectus aut autem
2 quis ut nam facilis et officia qui
3 fugiat veniam minus
4 et porro tempora
5 laboriosam mollitia et enim quasi adipisci quia provident illum
6 qui ullam ratione quibusdam voluptatem quia omnis
7 illo expedita consequatur quia in
8 quo adipisci enim quam ut ab
9 molestiae perspiciatis ipsa
10 illo est ratione doloremque quia maiores aut
11 vero rerum temporibus dolor
12 ipsa repellendus fugit nisi
13 et doloremque nulla
14 repellendus sunt dolores architecto voluptatum
15 ab voluptatum amet voluptas
16 accusamus eos facilis sint et aut voluptatem
17 quo laboriosam deleniti aut qui
```

3. Que hay en esta url ? https://jsonplaceholder.typicode.com/todos/2

https://petstore.swagger.io/v2/pet/findByStatus?status=available

PRACTICA P02. Pets

Estudia esta url, su contenido es XML però se puede leer como Json.

```
import requests
import json

response = requests.get('https://petstore.swagger.io/v2/pet/findByStatus?status=available')
# print(response.json())

for data in response.json():
    print(data)
```

Podrias listar unicamente las mascotas cuyo **`name`** sea distinto de **`doggie`** ?

https://httpbin.org/xml

PRACTICA P03. XML

Estudia esta url, su contenido es XML. Se puede leer como Json ?

Prueba con este código :

```
import xml.etree.ElementTree
response = requests.get("https://httpbin.org/xml")

string_xml = response.content
tree = xml.etree.ElementTree.fromstring(string_xml)

xml.etree.ElementTree.dump(tree)
```

https://www.washingtonpost.com/arcio/news-sitemap/

PRACTICA P04. Más a fondo.

https://gorest.co.in/public/v2/users

https://jsonplaceholder.typicode.com/todos/2

https://realpython.com/api-integration-in-python/