



Consumo de datos desde una API REST

Interactuando con una API REST

Implementar un aplicativo web que consume datos desde una API REST utilizando la librería Axios para dar solución a una problemática

- Unidad 1: Introducción a Componentes Web y Vue Js
- Unidad 2: Binding de formularios
- Unidad 3: Templates y rendering en Vue
- Unidad 4: Manejo de eventos y reutilización de componentes
- Unidad 5: Consumo de datos desde una API REST



Te encuentras aquí



¿Qué aprenderás en esta sesión?

- *Verifica el funcionamiento de una API Rest publicada en un servidor utilizando una herramienta cliente.*

¿Cómo se pueden obtener datos de una API REST con JavaScript?



/* Interactuando con una API REST */

Interactuando con una API REST

herramientas para probar API's

Existen varias herramientas para realizar pruebas rápidas a una API REST y revisar cuál es su **comportamiento** en diferentes escenarios.

Entre las herramientas populares tenemos las siguientes:

- Postman
- Insomnia
- Thunder Client (VSC)

Interactuando con una API REST

herramientas para probar API's

- Cualquiera de éstas herramientas nos servirán para operaciones básicas.
- Los usuarios eligen simplemente la que más le gusta.

A lo largo de esta unidad utilizaremos **POSTMAN** para probar la API **reqres**.



Ejercicio guiado

"Interactuando con una API REST"



Interactuando con una API REST

POSTMAN

1. Descarga POSTMAN de su página oficial ([enlace aquí](#)).

The image shows the Postman website's download page and a screenshot of the Postman application. The website has a navigation bar with links for Product, Pricing, Enterprise, Resources and support, and Explore. It includes 'Sign In' and 'Sign Up for Free' buttons. The main heading is 'Download Postman', followed by a paragraph: 'Download the app to get started using the Postman API Platform today. Or, if you prefer a browser experience, you can try the web version of Postman.' Below this, under 'The Postman app', is a 'Windows 64-bit' download button, which is pointed to by a red arrow. The application screenshot shows the 'Twitter's Public Workspace' with a 'GET Single Tweet' request selected. The request details show a GET method to the URL 'https://api.twitter.com/2/tweets/id'.

Product ▾ Pricing Enterprise ▾ Resources and support ▾ Explore

Sign In Sign Up for Free

Download Postman

Download the app to get started using the Postman API Platform today. Or, if you prefer a browser experience, you can try the web version of Postman.

The Postman app

Download the app to get started with the Postman API Platform.

Windows 64-bit

Home Workspaces ▾ API Network ▾ Explore

Search Postman

Twitter's Public Workspace New Import Overview

GET Single Tweet

... / Tweet Lookup / Request name Save ▾

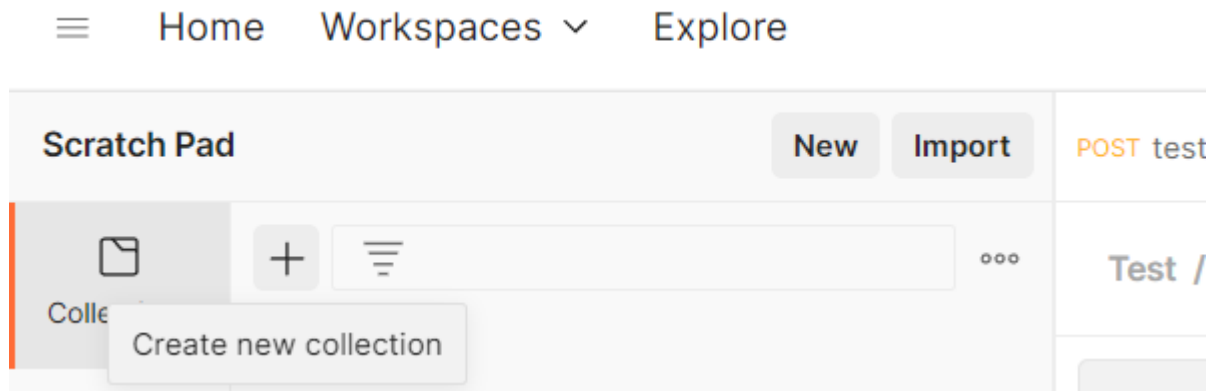
GET ▾ https://api.twitter.com/2/tweets/id Send

Params Authorization Headers Body Pre-request Scripts Tests Settings

Interactuando con una API REST

POSTMAN

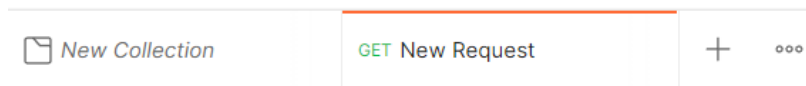
2. Abre POSTMAN en tu computadora y crea una nueva colección.





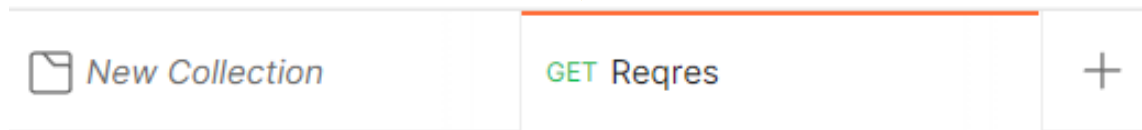
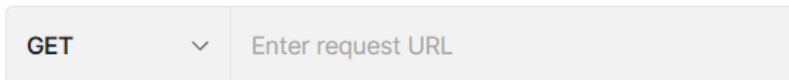
Interactuando con una API REST

POSTMAN

3. Guarda el nombre de la colección como **Reqres**.



New Collection / New Request  

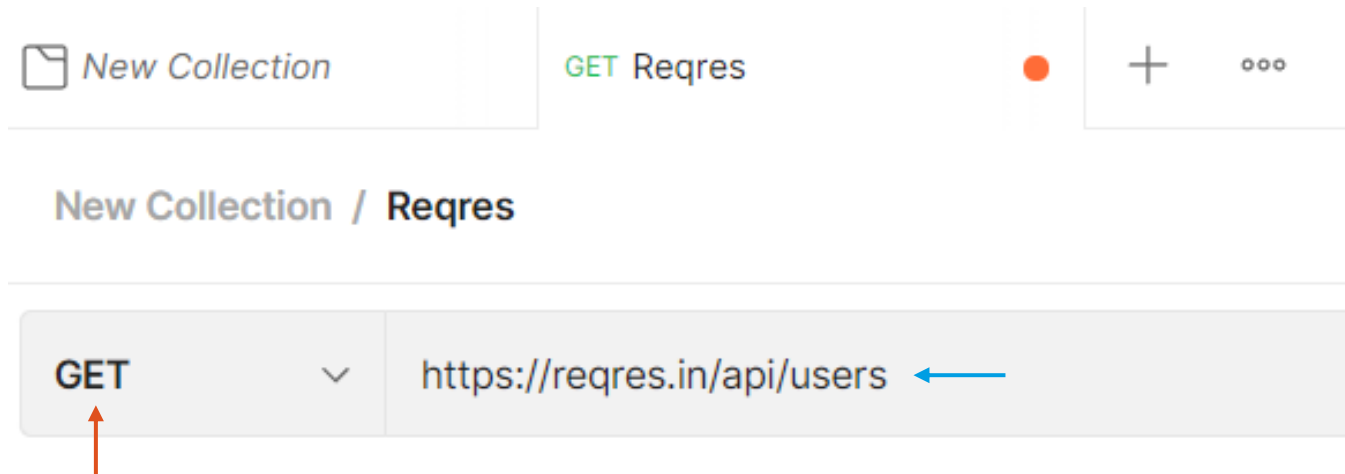


New Collection / Reqres

Interactuando con una API REST

POSTMAN

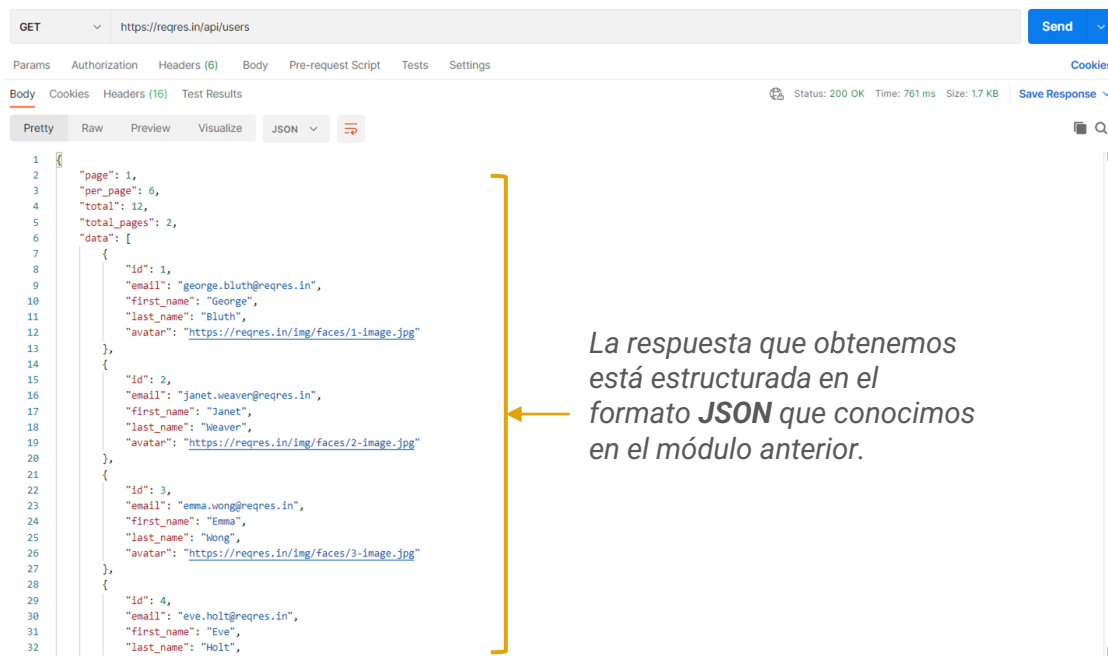
4. Escribe la URL <https://reqres.in/api/users> usando el método **GET**.



Interactuando con una API REST

POSTMAN

5. Envía la consulta y observa la respuesta obtenida.



The screenshot shows the Postman interface. At the top, a GET request is configured to `https://reqres.in/api/users`. A blue arrow points to the 'Send' button. Below the request bar, the 'Body' tab is selected, showing the response in 'Pretty' format. The response is a JSON object with a status of 200 OK, a time of 761 ms, and a size of 1.7 KB. A blue arrow points to the 'Save Response' button. The JSON response is as follows:

```
1 {
2   "page": 1,
3   "per_page": 6,
4   "total": 12,
5   "total_pages": 2,
6   "data": [
7     {
8       "id": 1,
9       "email": "george.bluth@reqres.in",
10      "first_name": "George",
11      "last_name": "Bluth",
12      "avatar": "https://reqres.in/img/faces/1-image.jpg"
13    },
14    {
15      "id": 2,
16      "email": "janet.weaver@reqres.in",
17      "first_name": "Janet",
18      "last_name": "Weaver",
19      "avatar": "https://reqres.in/img/faces/2-image.jpg"
20    },
21    {
22      "id": 3,
23      "email": "emma.wong@reqres.in",
24      "first_name": "Emma",
25      "last_name": "Wong",
26      "avatar": "https://reqres.in/img/faces/3-image.jpg"
27    },
28    {
29      "id": 4,
30      "email": "eve.holt@reqres.in",
31      "first_name": "Eve",
32      "last_name": "Holt",
```

La respuesta que obtenemos está estructurada en el formato **JSON** que conocimos en el módulo anterior.

Recordemos que el JSON es una notación que presenta la información con el formato:

{ propiedad : valor }

Todas las API's que utilizaremos por defecto retornan los datos en este formato



Interactuando con una API REST

POSTMAN

La respuesta que recibimos es data enviada desde el servidor de la api Reqres (reqres.in). En su página oficial encontrarás un apartado con las diferentes rutas (endpoints) que puedes consultar con sus métodos correspondientes.

GET	LIST USERS
GET	SINGLE USER
GET	SINGLE USER NOT FOUND
GET	LIST <RESOURCE>
GET	SINGLE <RESOURCE>
GET	SINGLE <RESOURCE> NOT FOUND
POST	CREATE
PUT	UPDATE

Request

`/api/users?page=2`

Ejercicio propuesto



Interactuando con una API REST

POSTMAN

- Usando POSTMAN, consulta la información del usuario de id 2 a través de otro endpoint de la API Reqres.

GET	LIST USERS
GET	SINGLE USER
GET	SINGLE USER NOT FOUND
GET	LIST <RESOURCE>
GET	SINGLE <RESOURCE>
GET	SINGLE <RESOURCE> NOT FOUND

Request
`/api/users/2`

Response
200

```
{
  "data": {
    "id": 2,
    "email": "janet.weaver@reqres.in",
    "first_name": "Janet",
    "last_name": "Weaver",
    "avatar": "https://reqres.in/img/fac"
  },
  "support": {
    "url": "https://reqres.in/#support-h",
    "text": "To keep ReqRes free, contri"
  }
}
```

/* Autenticación con JWT */

Autenticación con JWT

Cuando trabajamos con sistemas web de información privada o delicada, podemos encontrarnos con seguridad informática basada en **tokens JWT**.

Estos tokens son finalmente datos cifrados con una estructura determinada. Ejemplo:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
rgRG9lIiwiaXNTb2NpYWwiOnRydWV9.
4pcPyMD09o1PSyXnrXCjTwXyr4BsezDI1AVTmud2fU4

El encriptamiento de información puede realizarse tanto en el backend como en el frontend con librerías como JWT.

Sin embargo y por seguridad es preferible delegar esta tarea al servidor.

Los datos sensibles comúnmente encriptados son contraseñas e información privada de personas y empresas.

Autenticación con JWT

En esos casos donde interactuemos con API's que utilizan tokens podemos hacer pruebas en POSTMAN agregando una cabecera **Authorization** con un **token** como valor.

Para ésto, debes seleccionar:

GET ▼ https://reqres.in/api/users

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Headers 👁 6 hidden

	KEY	VALUE
<input checked="" type="checkbox"/>	Authorization ←	Bearer <token> ←

**Comparte con sus compañeros
screenshots de API's públicas
consultadas con POSTMAN**





Próxima sesión...

- *Guía de ejercicios*

{desafío}
latam_

*Academia de
talentos digitales*

