

Josseph Ernesto Paredes Rivera

Azael Bigvai Acuña Pineda

Diana Martinez

1. **Explora una API real** (10 min)

- Ve a: <https://jsonplaceholder.typicode.com/>

- Abre las herramientas de desarrollador (F12)

- Haz estas peticiones y observa:

- GET /posts (lista todos)
- GET /posts/1 (obtiene uno específico)
- POST /posts (crea nuevo - usa cualquier título/body)

****Anota**:** ¿Qué códigos HTTP ves? ¿Qué patrones observas en las URLs?

****Entregable**:** Documento con observaciones de la API y capturas de pantalla del navegador mostrando las peticiones.

1. GET /posts

- **Código HTTP:** 200 OK

- **Respuesta:** Lista de posts en formato JSON (array de objetos)

- **Patrón de URL:** /recurso (plural)

```
> fetch('https://jsonplaceholder.typicode.com/posts/1')
  .then(response => response.json())
  .then(json => console.log(json));
< ▶ Promise {<pending>}
▼ user: {
    id: 1,
    title: "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
    body: "quia et suscipit\\nsuscipit recusandae consequuntur \\nstrum rerum est autem sunt rem eveniet architecto"
  }
  id: 1
  title: "sunt aut facere repellat provident occaecati excepturi optio reprehenderit"
  userId: 1
  [ [Prototype] ]: Object
```

Name	Value	Headers	Preview	Response	Initiator	Timing	Cookies
rum		▼ General					
1		Request URL		https://jsonplaceholder.typicode.com/posts/1			
		Request Method		GET			
		Status Code		200 OK			
		Remote Address		104.21.16.1:443			
		Referrer Policy		strict-origin-when-cross-origin			
		▼ Response Headers					
		Access-Control-Allow-Credentials		true			
		Age		2675			
		Alt-Svc		h3="443"; ma=86400			
		Cache-Control		max-age=43200			
		CF-Cache-Status		HIT			
		CF-Ray		97cb3e6fc95bddd6-IAD			
		Content-Encoding		zstd			
		Content-Type		application/json; charset=utf-8			
		Date		Wed, 10 Sep 2025 01:45:05 GMT			
		Etag		W/"124-yiKdLzqOSgfBzJrcdJ8Yq0LNu"			
		Expires		-1			
		Nel		{"report_to": "heroku-nel", "response_headers": ["Vary"]}			
2 requests	1.4 kB transferred	292 B resources					

2. GET /posts

- **Código HTTP:** 200 OK
 - **Respuesta:** Un solo post (objeto JSON)
 - **Patrón de URL:** /recurso/{id}

The screenshot shows the Network tab in the Chrome DevTools Performance panel. A single network request for 'posts' is listed. The request URL is `https://jsonplaceholder.typicode.com/posts`, the method is GET, and the status code is 200 OK. The response body contains JSON data. The request took 20,000 ms.

Name	Headers	Preview	Response	Initiator	Timing	Cookies
rum	General					
1	Request URL					
posts	Request Method					
	Status Code					
	Remote Address					
	Referrer Policy					
	Response Headers					
	Access-Control-Allow-Credentials					
	Age					
	Alt-Svc					
	Cache-Control					
	CF-Cache-Status					
	CF-Ray					
	Content-Encoding					
	Content-Type					
	Date					
	Etag					
	Expires					

3. POST /posts

- **Código HTTP:** 201 Created

- **Respuesta:** Objeto creado con ID asignado

```
fetch('https://jsonplaceholder.typicode.com/posts', {
  method: 'POST',
  body: JSON.stringify({
    title: 'foo',
    body: 'bar',
    userId: 1,
  }),
  headers: {
    'Content-type': 'application/json; charset=UTF-8',
  }
})
.then((response) => response.json())
.then((json) => console.log(json));

▶ Promise {<pending>}
▼ {title: 'foo', body: 'bar', userId: 1, id: 101} ⓘ
  body: "bar"
  id: 101
  title: "foo"
  userId: 1
  ► [[Prototype]]: Object
```

- **Patrón de URL:** /recurso (plural)

The screenshot shows the Network tab in the Chrome DevTools developer console. A single network request is listed, which is a POST to the URL `https://jsonplaceholder.typicode.com/posts`. The request was successful, returning a status code of 201 Created. The response payload is a JSON object representing a new post with the properties: title: 'foo', body: 'bar', userId: 1, and id: 101. The Headers section of the response includes standard CORS headers like Access-Control-Allow-Origin and Cache-Control, along with a Location header pointing back to the same endpoint. The response body is also shown as a JSON object.

Name	Headers	Payload	Preview	Response	Initiator	Timing	Cookies
rum	General						
1							
posts							
posts	Response Headers						

Request URL: https://jsonplaceholder.typicode.com/posts
Request Method: POST
Status Code: 201 Created
Remote Address: 104.21.16.1:443
Referrer Policy: strict-origin-when-cross-origin

Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: https://jsonplaceholder.typicode.com
Access-Control-Expose-Headers: Location
Alt-Svc: h3=":443"; ma=86400
Cache-Control: no-cache
Cf-Cache-Status: DYNAMIC
Cf-Ray: 97cb4a214e3fd687-IAD
Content-Length: 65
Content-Type: application/json; charset=utf-8
Date: Wed, 10 Sep 2025 01:53:04 GMT
Etag: W/"41-GDNaWfnVU6RZhplbye0veBaqcHA"
Expires: -1

Fundamentos REST

REST

- Significa: Representational State Transfer
- Principio clave 1: Recursos identificados por URLs
- Principio clave 2: Uso de métodos HTTP estándar (GET, POST, PUT, DELETE)
- Principio clave 3: Stateless (sin estado entre peticiones)

Recursos vs Representaciones

- Recurso = Entidad conceptual (ej: un usuario, un producto)
- Representación = Forma concreta (ej: JSON, XML)
- Ejemplo: /users/123
 - El recurso es: el usuario con id 123
 - La representación es: el JSON devuelto

Códigos HTTP esenciales

- 200: OK (éxito)
- 201: Created (recurso creado)
- 400: Bad Request (petición inválida)
- 404: Not Found (no encontrado)
- 500: Internal Server Error (error del servidor)

Esquema de API REST para EcoMarket

Recurso principal: products

Método	Endpoint	Descripción
GET	/products	Lista todos los productos
GET	/products/{id}	Obtiene un producto específico

POST	/products	Crea un nuevo producto	
PUT	/products/{id}	Actualiza un producto	
DELETE	/products/{id}	Elimina un producto	

Justificación

- Se usa 'products' en plural siguiendo convención REST.
- PUT reemplaza todo el recurso, no crea si no existe (devuelve 404).
- DELETE devuelve 404 si el producto no existe.
- URLs son sustantivos, no verbos.
- Se consideran casos extremos (IDs inexistentes, datos inválidos).

Bitácora de Decisiones Arquitectónicas (EcoMarket)

- Recurso principal: products
- Endpoints: GET, POST, PUT, DELETE sobre /products y /products/{id}
- Validaciones: nombre no vacío, price ≥ 0 , stock ≥ 0
- Formato de errores: siempre JSON {"error": "mensaje"}
- PUT y DELETE devuelven 404 si el producto no existe
- Decisión difícil: PUT no crea, solo actualiza
- Mejoras futuras: paginación, filtros, autenticación, persistencia

Reflexión de Diseño

- Decisión más difícil: Elegir si PUT debe crear o solo actualizar. Decidimos que solo actualiza y devuelve 404 si no existe.
- Códigos HTTP: Elegimos 200 para éxito, 201 para creación, 400 para errores de validación, 404 para no encontrado, 204 para borrado exitoso.
- Como frontend developer, me gustaría que los errores incluyeran un campo "code" para facilitar el manejo en el cliente.