

Jossep 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>}
  ▼ {userId: 1, id: 1, title: 'sunt aut facere repellat provident occaecati excepturi optio reprehenderit', body: 'quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto'}
    id: 1
    title: "sunt aut facere repellat provident occaecati excepturi optio reprehenderit"
    userId: 1
    ► [[Prototype]]: Object
```

Name	⌵ Headers Preview Response Initiator Timing Cookies						
rum	⌵ General						
1	<div><div>Request URL</div><div>https://jsonplaceholder.typicode.com/posts/1</div></div> <div><div>Request Method</div><div>GET</div></div> <div><div>Status Code</div><div><div>200 OK</div></div></div> <div><div>Remote Address</div><div>104.21.16.1443</div></div> <div><div>Referrer Policy</div><div>strict-origin-when-cross-origin</div></div>						
	⌵ Response Headers						
	<div><div>Access-Control-Allow-Credentials</div><div>true</div></div> <div><div>Age</div><div>2675</div></div> <div><div>Alt-Svc</div><div>h3="443"; ma=86400</div></div> <div><div>Cache-Control</div><div>max-age=43200</div></div> <div><div>CF-Cache-Status</div><div>HIT</div></div> <div><div>CF-Ray</div><div>97cb3e6fc95bdda6-IAD</div></div> <div><div>Content-Encoding</div><div>zstd</div></div> <div><div>Content-Type</div><div>application/json; charset=utf-8</div></div> <div><div>Date</div><div>Wed, 10 Sep 2025 01:45:05 GMT</div></div> <div><div>Etag</div><div>W/"124-yikdLzqO5gB7rFrcdJB9q0LGnU"</div></div> <div><div>Expires</div><div>-1</div></div>						
2 requests 1.4 kB transferred 292 B resources	<div><div>Nel</div><div>["report_to":"heroku-nel","response_headers":["V]</div></div>						

2. GET /posts

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

```
> fetch('https://jsonplaceholder.typicode.com/posts')
.then((response) => response.json())
.then((json) => console.log(json));

• Promise (pending)

{
  "id": 1,
  "username": "yoshi",
  "email": "yoshi@yoshi.com",
  "password": "yoshi",
  "phone": "066-75737151",
  "website": "yoshi.com",
  "company": {
    "name": "Yoshi, Inc.",
    "catchPhrase": "Yoshi, Inc.",
    "bs": "Yoshi, Inc."
  }
}
```

20,000 ms

40,000 ms

60,000 ms

80,000 ms

100,000 ms

120,000 ms

140,000 ms

160,000 ms

180,000 ms

200,000 ms

220,000 ms

240,000 ms

Filter

Invert

More filters

Name

rum

1

posts

×

Headers

Preview

Response

Initiator

Timing

Cookies

▼ General

Request URL

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

Nel

3 requests 9.2 kB transferred 27.8 kB resources

["report_to","heroku-nel","response_headers"]

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 Chrome DevTools Network tab. A POST request to `https://jsonplaceholder.typicode.com/posts` is selected. The status is `201 Created`. The response headers are expanded, showing `Content-Type: application/json; charset=utf-8`.

Name	Value
rum	
1	
posts	
posts	

General	Value
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

Response Headers	Value
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

4 requests 10.1 kB transferred 27.9 kB resources

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.