



Nombre de Grupo 2 :

Adela Isabel Garay Alvarado

Leonardo Andres Velasque Castro

Diego Rual Funes Turcios

Clase: DESARROLLO DE APLICACIONES WEB I

Tema: Consumo de API de Terceros

Fecha: 20/6/2025

Explicación de los métodos y funciones utilizadas

- `express ()` y `app.use (express. json ())`

Se utiliza Express para crear el servidor de forma rápida y eficiente. El middleware `express. json ()` permite que las peticiones POST y PUT puedan leer y procesar datos en formato JSON.

- `axios`

Axios facilita hacer peticiones HTTP a la FakeStoreAPI de manera sencilla. Se usó en cada operación CRUD para comunicar el backend con la API externa.

- `app.get ('/productos', ...)`

Define una ruta GET para obtener todos los productos desde la API. Internamente, usa `axios.get ()` para hacer la solicitud y responde con los datos obtenidos en formato JSON.

- `app.post ('/productos', ...)`

Ruta POST que envía un nuevo producto a la API usando `axios.post ()`. Los datos se definen dentro del objeto nuevo Producto.

- `app.put ('/productos/:id', ...)`

Ruta para actualizar un producto existente. Se utiliza `axios.put ()` con la URL correspondiente al ID del producto, modificando al menos un campo (por ejemplo, el title).

- `app. delete ('/productos/:id', ...)`

Permite eliminar un producto usando `axios. delete ()` y el ID proporcionado en los parámetros de la URL.

- Ruta base /

Se agregó una ruta simple de prueba para verificar que el servidor está funcionando correctamente. Muestra un mensaje como “API de FakeStore funcionando correctamente”.

Desafíos encontrados y cómo los resolví

- Error al hacer una petición GET con un body JSON

Me aparecía un error porque estaba enviando datos en el cuerpo de una petición GET, lo cual no es necesario ni permitido. Lo resolví quitando el body en Postman para esa operación.

- Confirmar que el servidor estaba activo

Al principio no tenía una ruta raíz definida (`/`), lo cual dificultaba confirmar que el servidor respondía. Agregué una ruta de prueba que muestra un mensaje en el navegador o Postman.

- Validar que los endpoints de FakeStoreAPI acepten la estructura de datos correcta

Para el POST, tuve que asegurarme de usar los campos requeridos: title, price, description, category e image, tal como lo indica la documentación oficial.

```
1 const express = require('express');
2 const axios = require('axios');
3 const app = express();
4
5 app.use(express.json());
6
7 const API_URL = 'https://fakestoreapi.com/products';
8 app.get('/productos', async (req, res) => {
9   try {
10     const response = await axios.get(API_URL);
11     res.json(response.data);
12   } catch (error) {
13     res.status(500).send('Error al obtener productos');
14   }
15 });
16 app.post('/productos', async (req, res) => {
17   const nuevoProducto = {
18     title: 'Producto de prueba',
19     price: 29.99,
20     description: 'Descripción de prueba',
21     image: 'https://i.pravatar.cc',
22     category: 'electronics'
23   };
24
25   try {
26     const response = await axios.post(API_URL, nuevoProducto);
27     res.json(response.data);
28   } catch (error) {
29     res.status(500).send('Error al crear producto');
30   }
31 });
32 app.put('/productos/:id', async (req, res) => {
```

```
16 app.post('/productos', async (req, res) => {
31   });
32 app.put('/productos/:id', async (req, res) => {
33   const { id } = req.params;
34   const actualizacion = {
35     title: 'Producto actualizado'
36   };
37
38   try {
39     const response = await axios.put(`${API_URL}/${id}`, actualizacion);
40     res.json(response.data);
41   } catch (error) {
42     res.status(500).send('Error al actualizar producto');
43   }
44 });
45 app.delete('/productos/:id', async (req, res) => {
46   const { id } = req.params;
47
48   try {
49     const response = await axios.delete(`${API_URL}/${id}`);
50     res.json({ mensaje: 'Producto eliminado', data: response.data });
51   } catch (error) {
52     res.status(500).send('Error al eliminar producto');
53   }
54 });
55 app.listen(3000, () => {
56   console.log('Servidor corriendo en http://localhost:3000');
57 });
58 app.get('/', (req, res) => {
59   res.send('API de FakeStore funcionando correctamente');
60 });
61 app.get('/', (req, res) => {
```

```
32 app.put('/productos/:id', async (req, res) => {
39   const response = await axios.put(`${API_URL}/${id}`, actualizacion);
40   res.json(response.data);
41 } catch (error) {
42   res.status(500).send('Error al actualizar producto');
43 }
44 });
45 app.delete('/productos/:id', async (req, res) => {
46   const { id } = req.params;
47
48   try {
49     const response = await axios.delete(`${API_URL}/${id}`);
50     res.json({ mensaje: 'Producto eliminado', data: response.data });
51   } catch (error) {
52     res.status(500).send('Error al eliminar producto');
53   }
54 });
55 app.listen(3000, () => {
56   console.log('Servidor corriendo en http://localhost:3000');
57 });
58 app.get('/', (req, res) => {
59   res.send('API de FakeStore funcionando correctamente');
60 });
61 app.get('/', (req, res) => {
62   res.send('API de FakeStore funcionando correctamente');
63 });
64
65 app.listen(3000, () => {
66   console.log('Servidor corriendo en http://localhost:3000');
67 });
68
```

http://localhost:3000/productos

GET http://localhost:3000/productos

Send

Params Authorization Headers (8) Body Scripts Settings Cookies

Query Params


Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (8) Test Results

404 Not Found 59 ms 429 B

HTML Preview Visualize

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="utf-8">
5 <title>Error</title>
6 </head>
7 <body>
8 <pre>Cannot GET /productos%0A%0A%0A</pre>
9 </body>
10 </html>
```

 **http://localhost:3000/productos** Save Share

GET

http://localhost:3000/productos

Send

Params

Authorization

Headers (8)

Body

Scripts

Settings

Cookies

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSON

Beautify

```
1 {
2   "marca": "Toyota",
3   "modelo": "Corolla",
4   "año_fabricacion": 2022,
5   "estado": "activo"
6 }
7
```

Body

Cookies

Headers (8)

Test Results

404 Not Found

59 ms

429 B

🌐

⋮

HTML

Preview

Visualize

⋮

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="utf-8">
5 <title>Error</title>
6 </head>
7 <body>
8 <pre>Cannot GET /productos%0A%0A%0A</pre>
9 </body>
10 </html>
```

Postbot

Runner

Start Proxy

Cookies

Vault

Trash

⛶