Peticiones a Servicios Externos

Gabriel Rodríguez Flores

January 15, 2024

ADA ITS
INSTITUTO TECNOLÓGICO SUPERIOR

- Realizar peticiones a otras APIs

Contents

1 Teoría

1.1 Introducción a Peticiones a Servicios Externos

Cuando desarrollamos aplicaciones, a menudo necesitamos interactuar con servicios externos para obtener o enviar datos. Esto se logra a través de peticiones a APIs (Interfaces de Programación de Aplicaciones) y otros servicios web.

1.2 Bibliotecas Comunes para Peticiones

> A partir de la versión 18, Nodejs cuenta con el módulo `fetch` de forma nativa.

Existen varias bibliotecas en Node.js que facilitan la realización de peticiones HTTP. Algunas de las más utilizadas son:

- **`axios`:** Una biblioteca Promise-based que proporciona una interfaz fácil de usar para realizar peticiones HTTP.
- **`got`:** Una biblioteca liviana y rápida para realizar peticiones HTTP con soporte para streams y Promise.
- **`node-fetch`:** Un módulo que ofrece una interfaz de `window.fetch` compatible con Node.js.

2 Ejemplos

2.1 Uso de Axios

2.1.1 GET

Axios es ampliamente utilizado por su simplicidad y potencia. Aquí tienes un ejemplo básico:

```
1  const axios = require('axios');
2
3  async function getUserData() {
4    try {
5      const response = await axios.get('https://jsonplaceholder.typicode.
           com/users/1');
6      console.log('Datos del usuario:', response.data);
7    } catch (error) {
8      console.error('Error al obtener datos del usuario:', error.message)
           ;
9    }
10 }
```

```
11
12 getUserData();
```

2.1.2 POST

```
1  const axios = require('axios');
2
3  async function sendDataToApi() {
4    const dataToSend = {
5      // ... datos a enviar
6    };
7
8    try {
9      const response = await axios.post('https://jsonplaceholder.typicode
           .com/posts', dataToSend);
10     console.log('Respuesta de la API después de enviar datos:',
           response.data);
11   } catch (error) {
12     console.error('Error al enviar datos a la API:', error.message);
13   }
14 }
15
16 sendDataToApi();
```

2.2 Uso de Got

2.2.1 GET

Got es una opción eficiente y fácil de usar. Aquí hay un ejemplo:

```
1  const got = require('got');
2
3  async function getPostData() {
4    try {
5      const response = await got('https://jsonplaceholder.typicode.com/
           posts/1');
6      console.log('Datos del post:', response.body);
7    } catch (error) {
8      console.error('Error al obtener datos del post:', error.message);
9    }
10 }
11
12 getPostData();
```

2.2.2 POST

```
const got = require('got');

async function postDataToApi() {
  const dataToSend = {
    title: 'Nuevo Post',
    body: 'Contenido del nuevo post',
    userId: 1,
  };

  try {
    const response = await got.post('https://jsonplaceholder.typicode.com/posts', {
      json: dataToSend,
      responseType: 'json',
    });

    console.log('Respuesta de la API después de enviar datos:', response.body);
  } catch (error) {
    console.error('Error al enviar datos a la API:', error.message);
  }
}

postDataToApi();
```

3 Ejercicios

Buscar e integrar una API en nuestro servidor.

Fuentes: - RapidApi - Zylalabs - Public APIs by Anna - Public APIs Github - Public APIs by Sergio

Ideas: - Notion

4 Entregables

4.1 En clase

- Explorar la documentación de una API de tu elección.
- Crear un servicio que consuma una API

4.2 Tarea

- Crear una aplicación (servidor) simple que realice peticiones a esa API y muestre la información obtenida.
  - Una ruta GET que haga de puente entre el usuario y la API a consultar
  - Adaptar el uso de filtros, orden y paginado si procede