UTN FACULTAD REGIONAL CÓRDOBA
INGENIERÍA EN SISTEMAS DE INFORMACIÓN

Desarrollo de Software
Notas de Clase

Clase 23 - Librería Axios y React (*)

Axios es una librería de JavaScript muy popular para realizar solicitudes HTTP, y aunque no es específica para React, es utilizada a menudo en proyectos de React por varias razones:

1. Facilidad de uso: Axios ofrece una API sencilla y fácil de usar para realizar solicitudes HTTP, lo que facilita su integración con proyectos de React.
2. Soporte para navegadores y Node.js: Axios es compatible tanto con navegadores como con entornos Node.js, lo que lo hace versátil para el desarrollo de aplicaciones web y del lado del servidor.
3. Manejo de errores: Axios permite un manejo de errores más estructurado y fácil de entender, ya que utiliza promesas y captura errores de forma más explícita.
4. Interceptores: Axios permite interceptar solicitudes y respuestas antes de que sean manejadas por el código que las consume. Esto es útil para agregar funcionalidades adicionales, como la autenticación, el registro de errores, la modificación de encabezados, etc.
5. Cancelación de solicitudes: Axios proporciona una forma sencilla de cancelar solicitudes en curso, lo que puede ser útil en ciertos casos, como cuando el usuario navega fuera de una página antes de que se complete una solicitud.
6. Transformación de datos: Axios permite transformar los datos enviados y recibidos en una solicitud, lo que puede ser útil para la serialización y deserialización de datos, así como para la manipulación de datos antes de su procesamiento.
7. Configuración global: Axios permite establecer configuraciones globales para todas las solicitudes realizadas, lo que facilita la configuración de elementos comunes, como encabezados, tiempos de espera, etc.

Aunque React no necesita específicamente de Axios y puede utilizar otras bibliotecas o la API Fetch nativa de JavaScript, muchas personas optan por usar Axios debido a estas ventajas y a la amplia adopción en la comunidad de desarrolladores.

El sitio web de axios está en https://axios-http.com/

Por qué usar Axios en lugar de Fetch

Axios ofrece una experiencia de desarrollo más amigable y características adicionales en comparación con Fetch.

1. Mayor compatibilidad con navegadores antiguos.
2. Tratamiento automático de solicitudes y respuestas JSON.

3. Manejo de errores más sencillo y claro. Uso de interceptores para transformar y manejar solicitudes y respuestas.
4. Soporte para cancelación de solicitudes.
5. Monitoreo del progreso de carga y descarga.

Métodos más utilizados con axios

1. **axios.get(url[, config])**: Realiza una solicitud GET a la URL especificada. Puedes proporcionar una configuración opcional, como parámetros de consulta o encabezados personalizados.

2. **axios.post(url, data[, config])**: Realiza una solicitud POST a la URL especificada. Envía el objeto data en el cuerpo de la solicitud. También puedes proporcionar una configuración opcional, como encabezados personalizados.

3. **axios.put(url, data[, config])**: Realiza una solicitud PUT a la URL especificada. Envía el objeto data en el cuerpo de la solicitud. Puedes proporcionar una configuración opcional, como encabezados personalizados.

4. **axios.delete(url[, config])**: Realiza una solicitud DELETE a la URL especificada. Puedes proporcionar una configuración opcional, como parámetros de consulta o encabezados personalizados.

Axios proporciona características adicionales como interceptores, cancelación de solicitudes, transformación de datos y configuración global. Sin embargo, los métodos mencionados anteriormente son los más utilizados y fundamentales para realizar solicitudes HTTP con Axios.

Para consumir una URL con Axios en React, primero debes instalar la librería Axios en tu proyecto y luego importarla. A continuación, puedes utilizarla para realizar solicitudes HTTP dentro de tu componente React pero se recomienda crear un servicio independiente para manejar las llamadas API utilizando Axios y luego importar ese servicio en tus componentes React. Esto te permite mantener una separación de responsabilidades y hacer que tu código sea más reutilizable y fácil de mantener.

1. Instala Axios en tu proyecto:

```
npm install axios
```

2. Crea un archivo para el servicio, por ejemplo articulos.service.js:

```
import axios from 'axios';

const API_BASE_URL = 'https://dds.frc.utn.edu.ar/api'; //URL de ejemplo

async function BuscarTodos() {
try {
    const response = await axios.get(`${API_BASE_URL}/articulos`);
    return response.data;
} catch (error) {
    console.error('Error al obtener los datos:', error);
    throw error;
```

```
}
};

export const articulosService = {
    BuscarTodos
};
```

En este archivo, importamos Axios y exportamos una función buscarTodos que realiza una solicitud GET a la API utilizando directamente la función axios.get.

3. Importa el servicio en tu componente React y úsalo con el hook useEffect:

```
import React, { useState, useEffect } from 'react';
import { articulosService } from './services/articulos.service';

const BuscarArticulos = () => {
const [data, setArticulos] = useState(null);

useEffect(() => {
    async function BuscarArticulos() {
        let data = await articulosService.BuscarTodos();
        setArticulos(data);
    }
    BuscarArticulos();
}, []);

return (
    <div>
    {data ? (
        <table>
            <thead>
                <tr>
                    <th>Articulo</th>
                    <th>Precio</th>
                </tr>
            </thead>
            <tbody>
            {data.map((element) => (
                <tr key={element.IdArticulo}><td>{element.Nombre}</td>
                <td>{element.Precio}</td></tr>
                ))}
            </tbody>
        </table>
    ) : (
        <p>Cargando datos...</p>
    )}
    </div>
);
};

export default BuscarArticulos;
```

4. Para grabar con axios se utiliza el método post y para actualizar el método put. Por lo cual hay que agregar la siguiente función en el archivo articulos.service.js

```
    async function Grabar(item) {
    if (item.IdArticulo === 0) {
        await httpService.post(`${API_BASE_URL}/articulos`, item);
    } else {
        await
httpService.put(`${API_BASE_URL}/articulos/${item.IdArticulo}`, item);
    }
    }

    // exportar tambien la función Grabar
    export const articulosService = {
        Buscar,Grabar
    };
```

> En este ejempo el parámetro item es un objeto de javascript con la estructura de articulos.

5. Para borrar, es similar al anterior pero con el mètodo delete de axios:

```
async function Baja(item) {
    await
httpService.delete(`${API_BASE_URL}/articulos/${item.IdArticulo}`);
}

// exportar tambien la función Baja
export const articulosService = {
    Buscar,Baja,Grabar
};
```

Uso de interceptores con Axios

Los interceptores de Axios son funciones que se ejecutan antes o después de que una solicitud o respuesta pase por la instancia de Axios. Permiten modificar, transformar o manejar solicitudes y respuestas de manera más flexible antes de que lleguen a la aplicación o antes de ser enviadas al servidor. Los interceptores son útiles en una variedad de casos, como manejo de errores globales, agregar encabezados y monitorear el progreso de las solicitudes. Son una herramienta poderosa para modificar y manejar solicitudes y respuestas de forma centralizada y flexible, lo que facilita la implementación de lógica común en toda la aplicación.

Manejo de errores globales: Puedes usar interceptores para manejar errores de forma centralizada en todas las solicitudes y respuestas. Por ejemplo, si deseas mostrar una notificación cada vez que ocurra un error de red o un error de servidor, puedes hacerlo con un interceptor de respuesta:

```
axios.interceptors.response.use(
  (response) => {
```

```
    // Si la respuesta es exitosa, simplemente devuélvela
    return response;
  },
  (error) => {
    // Si hay un error, muéstralo en una notificación
    console.error("Se produjo un error:", error.message);
    // Rechazar la promesa con el error para que pueda ser manejado en la
aplicación
    return Promise.reject(error);
  }
);
```

Agregar headers: Si deseas agregar encabezados a todas las solicitudes, como un token de autenticación, puedes hacerlo con un interceptor de solicitud:

```
axios.interceptors.request.use((config) => {
  // Agregar el token de autenticación al encabezado 'Authorization'
  config.headers.Authorization = `Bearer ${localStorage.getItem("authToken")}`;
  return config;
});
```

Monitorear el progreso de las solicitudes: Si bien no es una característica de los interceptores en sí, Axios permite monitorear el progreso de las solicitudes y respuestas mediante la configuración de eventos. A continuación, se muestra un ejemplo de cómo monitorear el progreso de una solicitud de carga:

```
const config = {
  onUploadProgress: (progressEvent) => {
    const percentCompleted = Math.round((progressEvent.loaded * 100) /
progressEvent.total);
    console.log(`Progreso de carga: ${percentCompleted}%`);
  },
};

axios.post("/upload", data, config);
```