---1oct

en el lado back (ebay\_nodejs)

npm install cookie-parser - - save

cuando usas funciones middleware de sofware de tercero tienens q usar .use

para definir miiddleware tienes q usar el metodo use del obj express

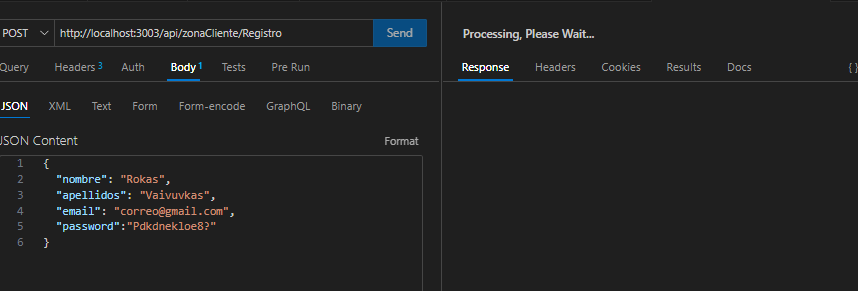
en el cookie-parser me da igual el endpoint solo quiero proccesar la peticion req y esta en la cabecera

en el body-parser me da igual el endpoint solo quiero extraer el request y el body

y ya apartir body paraser si me intenersa el endpoint get/post/put..

1 se lanza el servidor y luego el cluienet (react)

versiones modernas si hay vs anteriores con body y cokkieparser

en cabecera akades -> Cookie y mensaje='hola nodejs'  
en le body el cuerpo de la petición  
  
  
se queda pensando xq aqui no generamos ni respuestas ni invokas al next   
  
miServidorWeb.post('/api/zonaCliente/Registro',function(req,res,next){

console.log('el cliente de react ha hecho esta solicitud http-request...',req)

//console.log('datos del cliente(desde react) ',req.body); //conlsde de los datos q se mandan desde react

})  
  
meter esto  
 res.status(200).send({codifo:0 ,mensaje:'Has alcanzado bien el endpoint del registro cliente'})//generar respuest apara q no se quede pensando

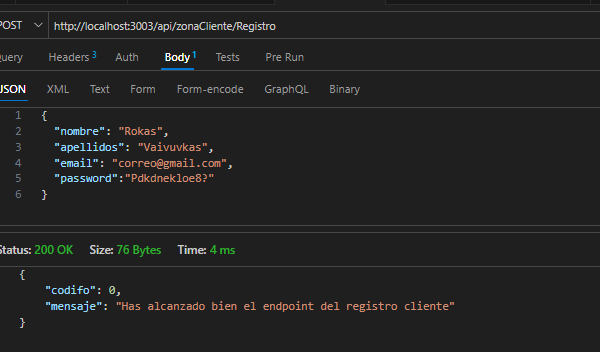
miServidorWeb.post('/api/zonaCliente/Registro',function(req,res,next){//no hace falta poner hhtp:localdpost :3009 xq estoy enell servidsor

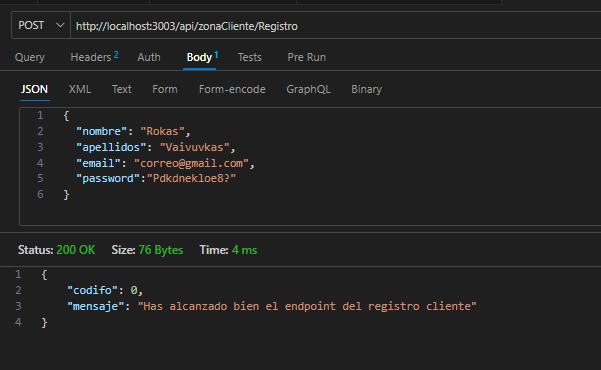
//console.log('el cliente de react ha hecho esta solicitud http-request...',req)

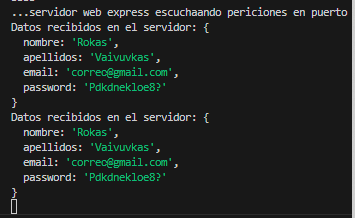
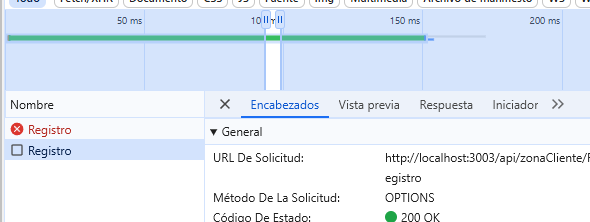
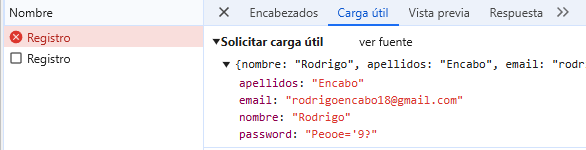
console.log('Datos del registro del ciente de react ...', req.body);//xq body parser los extrae de la solicitd del cliente del reac

res.status(200).send({codifo:0 ,mensaje:'Has alcanzado bien el endpoint del registro cliente'})//generar respuest apara q no se quede pensando

})

y sale lo mismo osea   
  
y klo datos x consola

bodyparser quitas el cabecera cooke  


y ves q se envia  
  
  
  
luego   
en rest service pones el puerto 3003 q es el del server   
  
y le das al boton de registrar   
enviar se envian pero  
  
  
  
sale un error del cors  
todos los navegadores tienen una política por defecto x seguridad-> tu como cliente solo puedes mandar datos ala misma url y mismo puerto q estas manejando no puedes cambiar ni el nombre del servidor ni el puerto osea q no te puedes salir .> y cuando invocas a node intentas acceder al 3003 y el navegador dice q estas intentado acceder aun sitio diferente con el q estas trabajando entonces no te deja   
entonces hay q habilitar cors   
todos los navegadores en el momento q detectan q cambias el puerto o nombre del servidor corta y no te deja

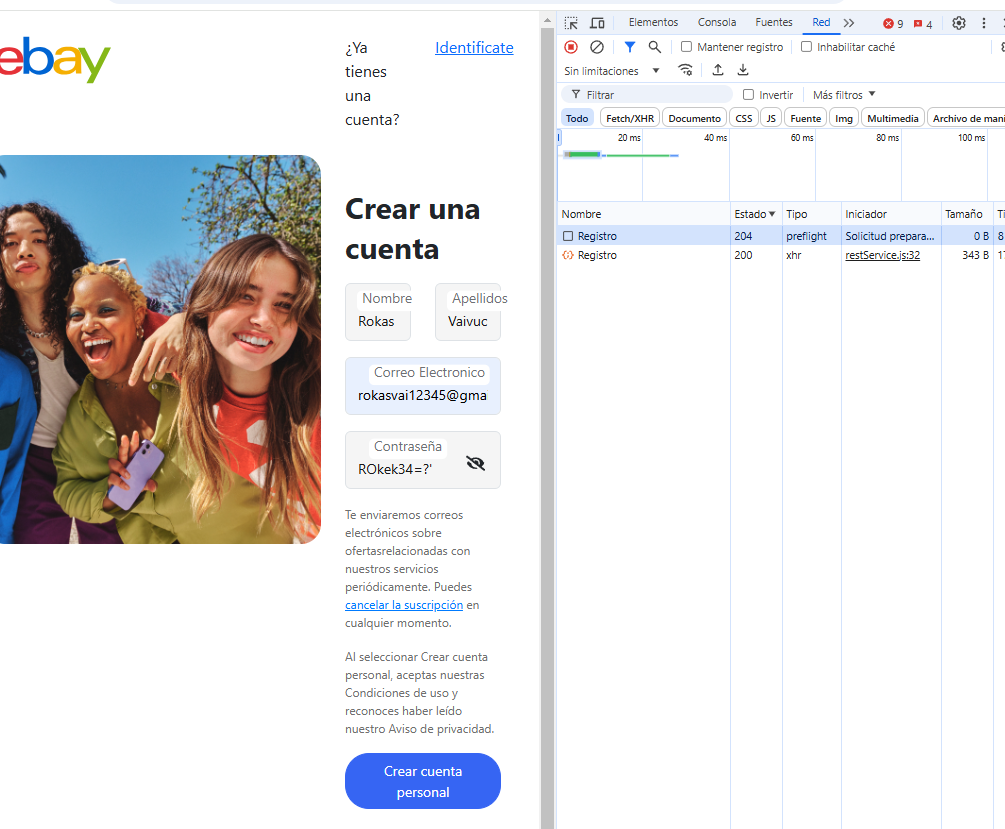
ves arriba q el metodo es options y no post -> antes de mandar los datos hace o intenta hacer una pregunta al servidor (request options request) -> q es oye admites solicitudes de clientes q no pertenecen a tu url ps el servidor dice no y el navegador corta > pero si el servidor dice q si automaticamente hace el POST -> entonces sempre hay como hos consultas un option y lo q quieras hacer  
  
pero para eso tienes q habilitar cors  
  
cors-> halita a clientes q no pertenecen a su mismo dominio (url) cualquiera q no le pertenezca a el pues le habilitar para q solicite informacion  
  
pregunta entrevista en front q es corn

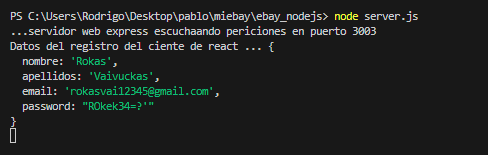
corn habita en el navegador o servidor

como funciona corcs

q hace el cliente antes de mandar los datos -> Option Request

entonces lo instalas npm install cors --save

luego lo improntas .:> conts cors =require(‘cors’)  
miServidorWeb.use(cors())  
metes datos y das al boton  


y ves que lo envia al servidor  


let \_respuesta = restService.RegistrarCliente({

nombre: formData.nombre.valor,

apellidos: formData.apellidos.valor,

email: formData.email.valor,

password: formData.password.valor

});

//como es asincrono este console log se lanza antes de recibir la respuests

//aerreglar pa q no salga undefinde q salgan l,os datos el servidor

console.log('datos recibidos del server NODEJS ante el registro :',\_respuesta);

imaginate q el restService (servicioq manda el cliemet) te dice el metodo registrarCliente q no te deja modificarllo osea solo datos el modifico metio callback como parametro   
ps hacerlo con eventos personalizados cualquier obj js puede disparar un evento

hay obj en q en vez de ejecutar otra funcion lo q hacen asu vez es disparar otro evento (una notificacion)   
el restservice cuando reciba los datos dela peticion ajax reciba un evento y ese evento interceptarlo en el cliente de react

solucion de JAVI  
// registro

restService.RegistrarCliente(

{

nombre: formData.nombre.valor,

apellidos: formData.apellidos.valor,

email: formData.email.valor,

password: formData.password.valor

},

function (error, \_respuesta) {

if (error) {

console.error("Error al enviar datos desde el registro", error);

} else {

console.log("Datos recibidos del server nodejs ante el registro", \_respuesta);

}

}

);

// restService

restService = {

RegistrarCliente: function (datoscuenta, callback) {

let \_petAjax = new XMLHttpRequest();

\_petAjax.open('POST', 'http://localhost:3003/api/zonaCliente/RegistrarCliente');

\_petAjax.setRequestHeader('Content-Type', 'application/json');

\_petAjax.addEventListener('readystatechange', function () {

if (\_petAjax.readyState === 4) {

if (\_petAjax.status >= 200 && \_petAjax.status < 300) {

// Llama al callback con el cuerpo de la respuesta en caso de éxito

callback(null, \_petAjax.responseText);

} else {

// Llama al callback con el error en caso de fallo en el estado

callback(new Error("Error en la solicitud: " + \_petAjax.statusText));

}

}

});

// Evento de error para problemas de conexión AJAX

\_petAjax.onerror = function () {

callback(new Error("Error en la conexión AJAX"));

};

// Envía los datos de la cuenta como JSON

\_petAjax.send(JSON.stringify(datoscuenta));

}

};

===============================

restService

cuando finaliza la petición de RegistrarCliente dispara un evento personalizado llamado "respuestaRegistro" con como datos del evento, van los datos de la petición (respuesta del servidor)

componente Registro.js de react añadir handler al evento "respuestaRegistro" del servicio restService (en el parámetro event del handler, estarán los datos de la respuesta del servidor)

promise  
Para solucionar el problema de la asincronía en ManejarSubmitForm, donde console.log se ejecuta antes de que la respuesta del servidor esté disponible, podemos modificar la función RegistrarCliente en restService.js para que devuelva una **promesa**. Esto permitirá a ManejarSubmitForm manejar la respuesta asincrónica con async/await. Aquí está cómo realizar los ajustes necesarios en ambos archivos.

### **1. Modificar restService.js para devolver una Promesa**

En restService.js, modifica RegistrarCliente para envolver la lógica AJAX en una Promesa. Cuando la petición AJAX se complete, la promesa se resolverá o se rechazará dependiendo de si fue exitosa o falló.

javascript

Copiar código

let restService = {

RegistrarCliente: function(datoscuenta) {

return new Promise((resolve, reject) => {

let \_petAjax = new XMLHttpRequest();

\_petAjax.open('POST', 'http://localhost:3003/api/zonaCliente/Registro');

\_petAjax.setRequestHeader('Content-Type', 'application/json');

\_petAjax.addEventListener('readystatechange', () => {

if (\_petAjax.readyState === 4) {

if (\_petAjax.status === 200) {

let \_respuestaServer = JSON.parse(\_petAjax.responseText);

resolve(\_respuestaServer); // Resuelve la promesa con la respuesta del servidor

} else {

reject('Error al registrar cliente: ' + \_petAjax.statusText); // Rechaza si hay un error en la respuesta

}

}

});

\_petAjax.send(JSON.stringify(datoscuenta));

});

},

// Otras funciones...

};

export default restService;

### **2. Modificar ManejarSubmitForm para manejar la Promesa con async/await**

Ahora que RegistrarCliente devuelve una promesa, podemos hacer que ManejarSubmitForm sea una función async y use await para esperar la respuesta de la promesa antes de continuar.

javascript

Copiar código

async function ManejarSubmitForm(ev) {

ev.preventDefault(); // Previene el envío predeterminado del formulario

try {

const \_respuesta = await restService.RegistrarCliente({

nombre: formData.nombre.valor,

apellidos: formData.apellidos.valor,

email: formData.email.valor,

password: formData.password.valor

});

// Este console.log se ejecutará después de recibir la respuesta del servidor

console.log('Datos recibidos del servidor NODEJS ante el registro:', \_respuesta);

} catch (error) {

console.error('Error en el registro:', error); // Manejo de errores en caso de fallo en la solicitud AJAX

}

}

### **Explicación de los Cambios**

1. **Promesa en restService.js**: Al envolver la lógica AJAX en una Promesa, permitimos que otras partes del código manejen el resultado una vez completado (con resolve o reject).
2. **async/await en ManejarSubmitForm**: Al convertir la función en async y usar await con RegistrarCliente, el código espera a que la respuesta esté disponible antes de continuar con console.log.

### **Resultado**

Este cambio garantiza que console.log en ManejarSubmitForm solo se ejecute después de recibir la respuesta del servidor, evitando así el undefined o la ejecución prematura del mensaje.