¿Qué es CORS?

CORS (Cross-Origin Resource Sharing) es un mecanismo de seguridad implementado en los navegadores que permite o restringe las solicitudes HTTP entre diferentes orígenes.

Por ejemplo, si tu frontend está en http://localhost:3000 y tu backend en http://localhost:4000, eso se considera 'cross-origin'.

¿Por qué existe CORS?

Los navegadores aplican la política del mismo origen (Same-Origin Policy) para proteger a los usuarios.

Esta política impide que scripts en una página web realicen peticiones a un dominio diferente sin permiso explícito. CORS es el mecanismo que da ese permiso.

¿Qué es CORS?

¿Cómo funciona CORS?

Cuando el navegador detecta una solicitud cross-origin, verifica si el servidor responde con los encabezados CORS adecuados.

Por ejemplo: Access-Control-Allow-Origin: http://localhost:3000

Hay dos tipos de solicitudes:

1. Simple requests (GET, POST con encabezados simples)
2. Preflight requests (usando OPTIONS)

¿Qué es CORS?

CORS en Node.js + Express

En Express se puede controlar CORS fácilmente con el paquete 'cors'.

Ejemplo básico:

```
const express = require('express');
const cors = require('cors');
const app = express();

app.use(cors()); // Permite todas las solicitudes

// O con configuración:
app.use(cors({ origin: 'http://localhost:3000' }));
```

¿Qué es CORS?

Errores comunes de CORS

- Access to fetch at 'http://localhost:4000/api' from origin 'http://localhost:3000' has been blocked by CORS policy.
- No 'Access-Control-Allow-Origin' header is present.

Estos errores indican que el servidor no está permitiendo solicitudes desde el frontend.

¿Qué es CORS?

Solucionando errores de CORS

1. Instalar el paquete cors:

   npm install cors

2. Usar app.use(cors()) o configurar orígenes permitidos.

3. Revisar los métodos y headers permitidos si usás solicitudes OPTIONS (preflight).

4. Siempre revisar que los dominios coincidan exactamente (incluyendo protocolo y puerto).

Tips útiles y casos especiales

- Podés usar un proxy en desarrollo para evitar problemas de CORS.
- Cuidado con permitir todos los orígenes (origin: '*') en producción.
- CORS no es un mecanismo de autenticación.
- Las cookies requieren configuración especial: credentials: 'include' + headers adecuados.

Resumen final

- CORS es clave para la seguridad entre dominios.
- En Node + Express se puede manejar fácilmente con el middleware 'cors'.
- Entender CORS te permite evitar errores frustrantes y construir APIs más seguras.

¡Ahora sí, a practicar! ->