



Universidad Autónoma de Nayarit



Unidad académica de economía

Carrera:

Sistemas computacionales

Materia:

Programación del lado del cliente

Tema:

Reto IA 8

Maestro:

Eligardo Cruz Sanches

Alumna:

Erika Alejandra Orozco Vázquez

Reto IA 8

Escenarios de prueba:

- Red lenta (latencia alta)
- Servidor intermitentemente disponible (falla cada 3ra petición)
- Respuesta truncada (conexión cortada a la mitad)
- Respuesta con formato inesperado (HTML en lugar de JSON)
- Timeout del servidor (respuesta después de 60 segundos)

Resultado:

```
erika@DESKTOP-I8L7BUH: ~/Semana1$ node cliente-corregido.js
🔥 Iniciando pruebas de creación...
Error al listar: API Error 404: <!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Error</title>
</head>
<body>
<pre>Cannot GET /productos</pre>
</body>
</html>

/home/erika/Semana1/Cliente-corregido.js:30
    throw new Error(`API Error ${respuesta.status}: ${body}`);
          ^
Error: API Error 404: <!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Error</title>
</head>
<body>
<pre>Cannot GET /productos</pre>
</body>
</html>

    at apiFetch (/home/erika/Semana1/Cliente-corregido.js:30:11)
    at process.processTicksAndRejections (node:internal/process/task_queues:95:5)
    at async listarProductos (/home/erika/Semana1/Cliente-corregido.js:41:17)

Node.js v18.19.1
erika@DESKTOP-I8L7BUH: ~/Semana1$ |
```

Cambios:

Se agregaron secciones de código para el manejo de:

*Reintentos automáticos:

```
async function apiFetch(endpoint, opciones = {}, timeout = 8000, reintentos = 2)
```

y cada que se da un error el número de reintentos se disminuye.

*Manejo de timeout con retry:

```
if (error.name === 'TimeoutError' || error.name === 'AbortError')
```

*Manejo especial de errores de red:

```
if (error.name === 'TypeError')
```

*Helper para parseo seguro:

```
async function procesarRespuestaJSON(respuesta)
```

```
if (!contentType.includes('application/json'))
```

Resultado:

```
erika@DESKTOP-I8L7BUH:~/Semanal$ node cliente-error.js
node:internal/modules/cjs/loader:1137
    throw err;
    ^

Error: Cannot find module '/home/erika/Semanal/cliente-error.js'
    at Module._resolveFilename (node:internal/modules/cjs/loader:1134:15)
    at Module._load (node:internal/modules/cjs/loader:975:27)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:128:12)
    at node:internal/main/run_main_module:28:49 {
  code: 'MODULE_NOT_FOUND',
  requireStack: []
}

Node.js v18.19.1
erika@DESKTOP-I8L7BUH:~/Semanal$ node cliente-errores.js
Iniciando pruebas de caos y creación...
Error al listar: API Error 404: <!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Error</title>
</head>
<body>
<pre>Cannot GET /productos</pre>
</body>
</html>

Producto 996 no encontrado.
Error al crear: API Error 404: <!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Error</title>
</head>
<body>
<pre>Cannot POST /productos</pre>
</body>
</html>
```

```
Error al crear: API Error 404: <!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Error</title>
</head>
<body>
<pre>Cannot POST /productos</pre>
</body>
</html>
```

```
erika@DESKTOP-I8L7BUH:~/Semanal$ const BASE_URL = 'http://localhost:3000';

/**
 * 1. FUNCIÓN CENTRALIZADA (Wrapper HTTP Resiliente)
 */
async function apiFetch(endpoint, opciones = {}, timeout = 8000, reintentos = 2) {
  const url = `${BASE_URL}${endpoint}`;

  const configuracion = {
    ...opciones,
    headers: {
      'Content-Type': 'application/json',
erika@DESKTOP-I8L7BUH:~/Semanal$ const BASE_URL = 'http://localhost:3000';
erika@DESKTOP-I8L7BUH:~/Semanal$ const BASE_URL = 'http://localhost:3000';

/**
 * 1. FUNCIÓN CENTRALIZADA (Wrapper HTTP Resiliente)
 */
async function apiFetch(endpoint, opciones = {}, timeout = 8000, reintentos = 2) {
  const url = `${BASE_URL}${endpoint}`;

  const configuracion = {
    ...opciones,
    headers: {
      'Content-Type': 'application/json',
      'Accept': 'application/json',
      'X-Client-Version': '1.0', // Conservamos tu header personalizado
    },
    ...opciones.headers
  },
  signal: AbortSignal.timeout(timeout)
}
```

```
/**
 * 1. FUNCIÓN CENTRALIZADA (Wrapper HTTP Resiliente)
 */
async function apiFetch(endpoint, opciones = {}, timeout = 8000, reintentos = 2) {
  const url = `${BASE_URL}${endpoint}`;

  const configuracion = {
    ...opciones,
    headers: {
      'Content-Type': 'application/json',
      'Accept': 'application/json',
      'X-Client-Version': '1.0', // Conservamos tu header personalizado
    },
    ...opciones.headers
  },
  signal: AbortSignal.timeout(timeout)
};

try {
  const respuesta = await fetch(url, configuracion);

  if (!respuesta.ok) {
    // Si el servidor falla (5xx), reintentamos
    if (respuesta.status >= 500 && reintentos > 0) {
      console.warn(`⚠ Error ${respuesta.status} en ${endpoint}. Reintentando... (${reintentos} restantes)`);
      await new Promise(res => setTimeout(res, 1000)); // Backoff de 1 segundo
      return apiFetch(endpoint, opciones, timeout, reintentos - 1);
    }

    // Si es otro error (ej. 400, 404), capturamos el texto sin romper el flujo
    const body = await respuesta.text().catch(() => "Cuerpo ilegible");
    throw new Error(`API Error ${respuesta.status}: ${body}`);
  }

  return respuesta;
} catch (error) {
  // Manejo de Timeout
  if (error.name === 'TimeoutError' || error.name === 'AbortError') {
    if (reintentos > 0) {
obtenerProductoPorId(996).catch(() => {});el precio como texto)ola no se ensucie si falla el mock
    }
  }
}
```

