

Diseñador de Capa de Logging

Muestra de Logs Reales

Se ejecutó una batería de pruebas simulando diferentes escenarios para validar la captura de logs en todos los niveles requeridos.

Captura de Evidencia: En la siguiente imagen se observan los logs generados en tiempo real desde la consola del navegador:

DEBUG (Gris): Muestra el inicio de la petición y la URL objetivo.

WARN (Naranja): Alerta automática disparada cuando la latencia supera los 2000ms.

ERROR (Rojo): Captura de excepciones críticas (ej. DNS no resuelto o errores 404).

```
--- INICIO DE PRUEBAS DE OBSERVABILIDAD ---                                     prueba_caos.html:99
[DEBUG] Iniciando GET https://retos-ia.free.beeceptor.com/api/productos prueba_caos.html:49
  ▶ {timestamp: '2026-01-31T05:25:22.711Z', level: 'DEBUG', message: 'Iniciando GET https://retos-ia.free.beeceptor.com/api/productos', url: 'https://retos-ia.free.beeceptor.com/api/productos', method: 'GET', ...}
[WARN] Respuesta lenta detectada (2437ms)                                         prueba_caos.html:49
  ▶ {timestamp: '2026-01-31T05:25:25.148Z', level: 'WARN', message: 'Respuesta Lenta detectada (2437ms)', url: 'https://retos-ia.free.beeceptor.com/api/productos', method: 'GET', ...}
[DEBUG] Iniciando POST https://httpstat.us/200?sleep=2500                         prueba_caos.html:49
  ▶ {timestamp: '2026-01-31T05:25:25.149Z', level: 'DEBUG', message: 'Iniciando POST https://httpstat.us/200?sleep=2500', url: 'https://httpstat.us/200?sleep=2500', method: 'POST', ...}
✖ ▶ POST https://httpstat.us/200?sleep=2500 net::ERR_EMPTY_RESPONSE               prueba_caos.html:64 ⓘ
[ERROR] Error de Red Crítico: Failed to fetch                                      prueba_caos.html:49
  ▶ {timestamp: '2026-01-31T05:25:25.551Z', level: 'ERROR', message: 'Error de Red Crítico: Failed to fetch', url: 'https://httpstat.us/200?sleep=2500', method: 'POST', ...}
[DEBUG] Iniciando GET https://httpstat.us/404                                       prueba_caos.html:49
  ▶ {timestamp: '2026-01-31T05:25:25.551Z', level: 'DEBUG', message: 'Iniciando GET https://httpstat.us/404', url: 'https://httpstat.us/404', method: 'GET', ...}
✖ ▶ GET https://httpstat.us/404 net::ERR_EMPTY_RESPONSE                           prueba_caos.html:64 ⓘ
[ERROR] Error de Red Crítico: Failed to fetch                                      prueba_caos.html:49
  ▶ {timestamp: '2026-01-31T05:25:25.992Z', level: 'ERROR', message: 'Error de Red Crítico: Failed to fetch', url: 'https://httpstat.us/404', method: 'GET', ...}
[DEBUG] Iniciando GET https://sitio-no-existente-xyz.com                            prueba_caos.html:49
  ▶ {timestamp: '2026-01-31T05:25:25.993Z', level: 'DEBUG', message: 'Iniciando GET https://sitio-no-existente-xyz.com', url: 'https://sitio-no-existente-xyz.com', method: 'GET', ...}
✖ ▶ GET https://sitio-no-existente-xyz.com/ net::ERR_NAME_NOT_RESOLVED          prueba_caos.html:64 ⓘ
[ERROR] Error de Red Crítico: Failed to fetch                                      prueba_caos.html:49
  ▶ {timestamp: '2026-01-31T05:25:26.068Z', level: 'ERROR', message: 'Error de Red Crítico: Failed to fetch', url: 'https://sitio-no-existente-xyz.com', method: 'GET', ...}
  > [ctrl] [i] to turn on code suggestions. Don't show again NEW
```

Propuesta de Mejoras para Versión 2.0

Respondiendo a la pregunta: "¿Qué información adicional te gustaría capturar y cómo mejorarías el sistema?", se proponen las siguientes dos evoluciones:

A. Implementación de Correlation IDs (Trazabilidad Distribuida)

Actualmente, si un usuario reporta un error, es difícil saber qué ocurrió exactamente en el servidor.

Mejora: Generar un X-Correlation-ID (UUID único) en el cliente al inicio de cada petición.

Beneficio: Este ID se envía al Backend y se registra en todos los logs (Frontend, API, Base de Datos). Esto permite rastrear una transacción completa "punta a punta" filtrando por un solo ID.

B. Persistencia y Agregación en la Nube

Actualmente, los logs solo viven en la consola del navegador del usuario ("Console.log"), por lo que se pierden al cerrar la pestaña.

Mejora: Crear un mecanismo de "Batching" (lotes) que acumule logs en un array y los envíe cada 30 segundos a un servicio externo (como Datadog, Sentry o ELK Stack).

Beneficio: Permite al equipo de ingeniería visualizar dashboards en tiempo real y recibir alertas automáticas (ej. Slack/Email) cuando la tasa de errores sube repentinamente, sin esperar a que los usuarios se quejen.