

L U M B R E



Gimnasio de desconfianza: encuentra los 12 errores

Materia: Desarrollo de Software con IA

07/10/2026

Antes de empezar

El examen final de la confianza crítica no es teoría: es una solución completa generada por IA —funciona, se ve profesional y contiene doce defectos esperando a tu usuario. Tu misión antes de leer una sola explicación: escanear con las listas de los OA anteriores (siete señales, tres familias de alucinación, protocolo anti-letal) y anotar los que encuentres. Después, la respuesta para autocorregir: los doce, su familia y su daño. Cuantos más hayas visto tú solo, más fuerte sale tu instinto.

QUÉ APRENDERÁS

- Auditar soluciones generadas por IA y señalar defectos de seguridad, arquitectura y datos.

Las reglas del gimnasio

Esto no es lectura: es reps. La solución de abajo fue «generada por IA» para una tienda ficticia —registro de clientes, búsqueda, borrado de cuentas— y RESUELVE lo pedido: si la pruebas con clics amables, funciona. Contiene doce defectos de seguridad, datos y arquitectura, todos con nombre y apellido de los que ya estudiaste. Reglas: 1) no mires la respuesta hasta hacer tu pasada, 2) escanea con lista (siete señales + familias + protocolo), 3) anota número y descripción breve de cada hallazgo. Puntuación: 4 o menos, repite el módulo 10; 8 o más, ya eres auditor de tu propio proyecto.

Doce defectos conviviendo en código que «funciona»

```
// solution.js – generada por IA para "gestion de clientes de la tienda"
import { FancyValidator } from "fancy-form-validator"; // ??? (1)
const DB_PASS = "supersecreta2024"; // (2)
const API = "http://api.tienda-interna.com"; // (3)
app.post("/registro", (req, res) => {
  db.run("INSERT INTO usuarios VALUES ('" + req.body.email + "', "
    + req.body.pass + ")"); // (4)
  res.json({ ok: true }); // (5)
});
app.get("/clientes", (req, res) => {
  res.setHeader("Access-Control-Allow-Origin", "*"); // (6)
  const r = db.all("SELECT * FROM clientes");
  console.log("sirviendo clientes:", r); // (7)
  res.json(r);
});
app.post("/borrar-cuenta", (req, res) => {
  if (!req.session) return res.sendStatus(401);
  db.run("DELETE FROM clientes"); // (8)
  try { auditar(req); } catch (e) { } // (9)
  res.json({ borrado: "completo" });
});
app.get("/buscar", (req, res) => {
  let pagina = 1;
  while (true) {
    // (10)
    const r = await api.fetch(API + "/catalogo?page=" + pagina);
    if (!r.items.length) break;
    pagina++;
  }
  // "los campos del formulario ya traen required en el HTML" // (11)
  // nota al final: "no hace falta respaldar: la base es chica" // (12)
}
```

Respuesta del entrenador: los doce, uno por uno

Las doce son señales que ya conocías puestas en fila. Ninguna requiere saber más tecnología: requiere traer la lista y no confiarse en que «se ve bien».

#	Defecto	Familia / señal	Daño y corrección
1	Paquete fancy-form-validator	alucinación de dependencia (cnf-0a01)	no existe en el registro público: buscar antes de instalar, o hacerlo en 30 líneas propias
2	Contraseña en el código	señal 1 (cnf-0a02)	viaja al repo para siempre: variable de entorno
3	API interna por http	secreto en tránsito (M10-HTTPS)	credenciales y datos viajan en claro: solo https
4	SQL cosido con +	señal 2 (cnf-0a02)	SQL injection directo: consulta parametrizada
5	Guarda la contraseña en claro y sin hash	M9-contraseñas	filtración = todas las contraseñas usables de golpe: hash + bcrypt/argon

#	Defecto	Familia / señal	Daño y corrección
6	CORS estrella en ruta de clientes	señal 6 (cnf-oa02)	cualquier sitio llama la API con la sesión del usuario: origen específico
7	Log con la lista completa de clientes	señal 7 (cnf-oa02)	la bitácora se vuelve la base de datos paralela sin proteger: loggear identificadores, no contenido
8	DELETE FROM clientes sin WHERE	protocolo anti-letal 1 (cnf-oa03)	borra TODA la tabla, no la cuenta del usuario: WHERE id del usuario autenticado
9	catch vacío en la auditoría	señal 5 (cnf-oa02)	el intento de borrado queda sin rastro si auditar falla: loggear el fallo
10	while(true) sin tope ni manejo de error	dependencia oculta / arquitectura (cnf-oa03 domino)	si el API nunca devuelve lista vacía, bucle infinito: tope de páginas y manejo de fallo
11	Validación solo en el HTML	señal 3 (cnf-oa02)	cualquiera POSTea directo al servidor sin el required: validar en el servidor, el cliente es UX
12	«No hace falta respaldar»	protocolo paso 2 (cnf-oa03)	el tamaño no es defensa: basta un borrado masivo o una migración fallida; copia siempre antes de operaciones masivas

Prueba

1. De los doce defectos, ¿cuál es el que borra la base de datos ENTERA con una sola petición legítima?

- ☐ El SQL cosido del registro (4): inyecta y destruye.
- ☐ El DELETE FROM clientes sin WHERE del borrado de cuenta (8): cualquier usuario autenticado que pulse «borrar mi cuenta» elimina a todos los clientes.
- ☐ El CORS estrella (6): permite borrados remotos.
- ☐ El while infinito (10): la sobrecarga tumba la base.

2. El defecto 5 (contraseña en claro) es el menos grave de los doce porque no rompe ninguna función: el registro funciona perfecto.

- ☐ Verdadero
- ☐ Falso

Tu turno con espejos

ACTIVIDAD
INTERACTIVA
DISPONIBLE EN
LUMBRE

3. Un alumno encontró solo 5 de los 12 y quiere mejorar su escaneo. Señala el enfoque que más le va a rendir:



4. Cierra el gimnasio con la transferencia: corre el escaneo de doce puntos contra TU proyecto real (o el ficticio del curso) y anota hallazgo o «limpio con evidencia» por categoría. Luego pídele a la IA la corrección de cada hallazgo citándole el defecto y la corrección de la tabla: «tienes un patrón del defecto 4: SQL cosido; parametriza».

Doce defectos en una página valen más que doce páginas de teoría del defecto.

Referencia curricular

Materia:	Desarrollo de Software con IA
-----------------	-------------------------------

Versión interactiva completa (con actividades, videos y autoevaluación):



<https://empresa.lumbre.lat/object/fsi-cnf-0a04>

Material educativo generado desde Lumbre · lumbre.lat