AUDITORIA TECNICA

ANNALOGICA

Analisis Tecnico Detallado con Codigo

Fecha: 2025-10-10

Version: 1.0.0

Tipo: Documento Tecnico

RESUMEN TECNICO

14 problemas criticos identificados
Esfuerzo estimado: 50-60 horas desarrollo
Tiempo total: 3-4 semanas (con legal)
Prioridad maxima: GDPR + Seguridad

1. SEGURIDAD - PROBLEMA CRITICO #1

Tokens JWT en localStorage (Vulnerable a XSS)

UBICACION: app/api/auth/login/route.ts:55

RIESGO: Si hay vulnerabilidad XSS, el atacante puede robar tokens.

Codigo Actual (Vulnerable):

```
// ACTUAL (INSEGURO):
return NextResponse.json({
  token,  // Token expuesto en JavaScript
  user: { id: user.id, email: user.email }
});
```

Solucion Recomendada:

```
// RECOMENDADO (SEGURO):
const response = NextResponse.json({
  user: { id: user.id, email: user.email }
});

response.cookies.set('auth_token', token, {
  httpOnly: true,      // No accesible desde JavaScript
  secure: true,        // Solo HTTPS
  sameSite: 'strict',  // Proteccion CSRF
  maxAge: 7 * 24 * 60 * 60, // 7 dias
  path: '/'
});

return response;
```

CAMBIOS EN FRONTEND:

- Eliminar localStorage.setItem("token")
- Cookies se envian automaticamente en requests
- Verificacion en middleware de Next.js

Middleware para Rutas Protegidas:

```
// middleware.ts (NUEVO)
import { NextResponse } from 'next/server';
import type { NextRequest } from 'next/server';
import jwt from 'jsonwebtoken';

export function middleware(request: NextRequest) {
  const token = request.cookies.get('auth_token')?.value;

  if (!token) {
    return NextResponse.redirect(new URL('/login', request.url));
  }

  try {
    jwt.verify(token, process.env.JWT_SECRET!);
    return NextResponse.next();
  } catch {
    return NextResponse.redirect(new URL('/login', request.url));
  }
}

export const config = {
  matcher: ['/dashboard/:path*', '/settings/:path*']
};
```

IMPACTO: Alto | PRIORIDAD: CRITICA | TIEMPO: 2 horas

2. SEGURIDAD - PROBLEMA CRITICO #2

Sin Content Security Policy (CSP)

UBICACION: next.config.ts:3

RIESGO: Code injection, clickjacking, MIME sniffing attacks.

Configuracion Actual:

```
// ACTUAL (NO EXISTE):
const nextConfig: NextConfig = {
  /* config options here */
};

export default nextConfig;
```

Solucion Completa (next.config.ts):

```
// RECOMENDADO:
import type { NextConfig } from "next";

const nextConfig: NextConfig = {
  async headers() {
    return [
      {
        source: '/:path*',
        headers: [
          // Previene clickjacking
          {
            key: 'X-Frame-Options',
            value: 'DENY'
          },
          // Previene MIME sniffing
          {
            key: 'X-Content-Type-Options',
            value: 'nosniff'
          },
          // Referrer policy
          {
            key: 'Referrer-Policy',
            value: 'strict-origin-when-cross-origin'
          },
          // Permissions policy
          {
            key: 'Permissions-Policy',
            value: 'camera=(), microphone=(), geolocation=()'
          },
          // Content Security Policy
          {
            key: 'Content-Security-Policy',
            value: [
              "default-src 'self'",
              "script-src 'self' 'unsafe-eval' 'unsafe-inline'",
              "style-src 'self' 'unsafe-inline'",
              "img-src 'self' data: https:",
              "font-src 'self'",
              "connect-src 'self' https://api.anthropic.com https://api.assemblyai.com",
              "frame-ancestors 'none'"
```

IMPACTO: Alto | PRIORIDAD: CRITICA | TIEMPO: 4 horas

3. INFRAESTRUCTURA - PROBLEMA CRITICO #3

Sin Circuit Breaker para APIs Externas

UBICACION: lib/assemblyai-client.ts:44-94

RIESGO: Si AssemblyAI/Claude fallan, todas las transcripciones fallan.

Dependencia Necesaria:

```
// INSTALAR:
npm install opossum
```

Implementacion Circuit Breaker:

```
// lib/circuit-breakers.ts (NUEVO)
import CircuitBreaker from 'opossum';
import { transcribeAudio } from './assemblyai-client';

const options = {
  timeout: 60000,              // 60s timeout
  errorThresholdPercentage: 50, // Abre si 50% fallan
  resetTimeout: 30000,         // Reintentar despues de 30s
  volumeThreshold: 5           // Min requests antes de abrir
};

export const assemblyAIBreaker = new CircuitBreaker(
  transcribeAudio,
  options
);

// Fallback cuando circuito abierto
assemblyAIBreaker.fallback(() => ({
  error: 'AssemblyAI temporalmente no disponible. Reintenta en 30s.'
}));

// Eventos para logging
assemblyAIBreaker.on('open', () => {
  console.error('[Circuit Breaker] OPEN - AssemblyAI fallando');
  // Enviar alerta critica
});

assemblyAIBreaker.on('halfOpen', () => {
  console.warn('[Circuit Breaker] HALF-OPEN - Probando AssemblyAI');
});

assemblyAIBreaker.on('close', () => {
  console.log('[Circuit Breaker] CLOSED - AssemblyAI recuperado');
});
```

Uso en Inngest Functions:

```
// lib/inngest/functions.ts - MODIFICAR:
import { assemblyAIBreaker } from '@/lib/circuit-breakers';

// Cambiar llamada directa por circuit breaker:
const transcriptionResult = await step.run('transcribe-audio', async () => {
  return await assemblyAIBreaker.fire({
    audioUrl,
    language: 'es',
    speakerLabels: true
  });
});
```

IMPACTO: Alto | PRIORIDAD: CRITICA | TIEMPO: 6 horas

4. GESTION COSTOS - PROBLEMA CRITICO #4

Sin Quotas Mensuales por Usuario

UBICACION: lib/db.ts:3-132 (falta campo en users)

RIESGO: Usuarios pueden consumir recursos ilimitados.

Migracion SQL:

```
-- Ejecutar en Neon Postgres Console:

-- 1. Agregar campos de quota
ALTER TABLE users
ADD COLUMN IF NOT EXISTS monthly_transcription_quota INT DEFAULT 100,
ADD COLUMN IF NOT EXISTS transcriptions_used_this_month INT DEFAULT 0,
ADD COLUMN IF NOT EXISTS quota_reset_date TIMESTAMP DEFAULT NOW();

-- 2. Indice para queries frecuentes
CREATE INDEX IF NOT EXISTS idx_users_quota_check
ON users(id, transcriptions_used_this_month, monthly_transcription_quota);

-- 3. Funcion para resetear quotas mensualmente
CREATE OR REPLACE FUNCTION reset_monthly_quotas()
RETURNS void AS $$
BEGIN
  UPDATE users
  SET transcriptions_used_this_month = 0,
      quota_reset_date = CURRENT_TIMESTAMP
  WHERE quota_reset_date < NOW() - INTERVAL '1 month';
END;
$$ LANGUAGE plpgsql;

-- 4. Comentarios
COMMENT ON COLUMN users.monthly_transcription_quota
  IS 'Numero maximo de transcripciones permitidas por mes';
COMMENT ON COLUMN users.transcriptions_used_this_month
  IS 'Contador de transcripciones usadas en el mes actual';
```

Codigo TypeScript (lib/db.ts):

```
// lib/db.ts - ACTUALIZAR INTERFACE:

export interface User {
  id: string;
  email: string;
  password: string;
  created_at: Date;
  updated_at: Date;
  // NUEVOS CAMPOS:
  monthly_transcription_quota: number;
  transcriptions_used_this_month: number;
  quota_reset_date: Date;
}

// NUEVA FUNCION:
export const UserDB = {
  // ... funciones existentes ...

  // Verificar y consumir quota
  consumeQuota: async (userId: string): Promise<{
    allowed: boolean;
    remaining: number;
    resetDate: Date;
  }> => {
    const user = await UserDB.findById(userId);
    if (!user) {
      throw new Error('Usuario no encontrado');
    }

    // Verificar si necesita reset mensual
    const monthAgo = new Date();
    monthAgo.setMonth(monthAgo.getMonth() - 1);

    if (user.quota_reset_date < monthAgo) {
      await sql`
        UPDATE users
        SET transcriptions_used_this_month = 0,
            quota_reset_date = CURRENT_TIMESTAMP
        WHERE id = ${userId}
      `;
```

Uso en API de Procesamiento:

```
// app/api/process/route.ts - AGREGAR VERIFICACION:

export async function POST(request: Request) {
  try {
    const user = verifyRequestAuth(request);
    if (!user) {
      return Response.json({ error: 'No autorizado' }, { status: 401 });
    }

    // NUEVO: Verificar quota ANTES de procesar
    const quotaCheck = await UserDB.consumeQuota(user.userId);

    if (!quotaCheck.allowed) {
      return Response.json({
        error: 'Cuota mensual agotada',
        remaining: quotaCheck.remaining,
        resetDate: quotaCheck.resetDate.toISOString(),
        message: 'Has alcanzado tu limite de ' +
                 'transcripciones este mes. Actualiza tu plan o ' +
                 'espera hasta: ' + quotaCheck.resetDate.toLocaleDateString()
      }, { status: 402 }); // 402 Payment Required
    }

    // ... resto del codigo de procesamiento ...

    // En respuesta exitosa, informar quota restante:
    return Response.json({
      success: true,
      jobId: job.id,
      status: 'pending',
      quotaRemaining: quotaCheck.remaining,
      quotaResetDate: quotaCheck.resetDate
    });

  } catch (error: any) {
    console.error('[API Process] Error:', error);
    return Response.json(
      { error: error.message },
      { status: 500 }
    );
```

Cron Job Mensual:

```
// Cron job para reset mensual (vercel.json):
{
  "crons": [
    {
      "path": "/api/cron/cleanup",
      "schedule": "0 2 * * *"
    },
    {
      "path": "/api/cron/reset-quotas",
      "schedule": "0 0 1 * *"
    }
  ]
}

// app/api/cron/reset-quotas/route.ts (NUEVO):
import { sql } from '@vercel/postgres';

export async function GET(request: Request) {
  const authHeader = request.headers.get('authorization');

  if (authHeader !== `Bearer ${process.env.CRON_SECRET}`) {
    return Response.json({ error: 'Unauthorized' }, { status: 401 });
  }

  try {
    const result = await sql`
      UPDATE users
      SET transcriptions_used_this_month = 0,
          quota_reset_date = CURRENT_TIMESTAMP
```

IMPACTO: Alto | PRIORIDAD: CRITICA | TIEMPO: 4 horas

5. OBSERVABILIDAD - PROBLEMA CRITICO #5

Sin Sistema de Monitoreo (Solo console.log)

RIESGO: Errores criticos no detectados, sin alertas automaticas.

Setup Sentry:

```
// Instalar dependencias:
npm install @sentry/nextjs
```

Configuracion Sentry (3 archivos):

```
// sentry.client.config.ts (NUEVO)
import * as Sentry from "@sentry/nextjs";

Sentry.init({
  dsn: process.env.NEXT_PUBLIC_SENTRY_DSN,
  environment: process.env.NODE_ENV,
  tracesSampleRate: 0.1, // 10% de requests
  replaysSessionSampleRate: 0.1,
  replaysOnErrorSampleRate: 1.0,
});

// sentry.server.config.ts (NUEVO)
import * as Sentry from "@sentry/nextjs";

Sentry.init({
  dsn: process.env.NEXT_PUBLIC_SENTRY_DSN,
  environment: process.env.NODE_ENV,
  tracesSampleRate: 0.1,
});

// sentry.edge.config.ts (NUEVO)
import * as Sentry from "@sentry/nextjs";

Sentry.init({
  dsn: process.env.NEXT_PUBLIC_SENTRY_DSN,
  tracesSampleRate: 0.1,
});
```

Integracion en Codigo:

```
// lib/inngest/functions.ts - AGREGAR ERROR TRACKING:

import * as Sentry from '@sentry/nextjs';

export const transcribeFile = inngest.createFunction(
  { ... },
  { event: 'task/transcribe' },
  async ({ event, step }) => {
    const { jobId } = event.data;

    try {
      // ... codigo existente ...

    } catch (error: any) {
      // CAPTURAR ERROR EN SENTRY:
      Sentry.captureException(error, {
        tags: {
          jobId,
          userId: event.data.userId,
          function: 'transcribeFile'
        },
        contexts: {
          job: {
            id: jobId,
            filename: event.data.filename,
```

CONFIGURACION GRATUITA: 10,000 eventos/mes (suficiente para inicio)

IMPACTO: Medio-Alto | PRIORIDAD: ALTA | TIEMPO: 4 horas