Informe SonarQube - Resumen de Issues (03/06/2025)

Este documento resume los hallazgos del primer análisis de código realizado con SonarQube, detallando los bugs, security hotspots y code smells identificados, junto con las soluciones propuestas.

1. Bugs (Errores)

a. object-fit duplicado

- **Descripción:** La propiedad object-fit está duplicada en el elemento .icon.
- **Ubicación:** client/src/components/Title/Title.module.css (línea 19)
- **Solución:** Eliminar la línea duplicada de object-fit:contain;.

```
/* Antes */
.icon {
    height: 8.9em;
    object-fit: contain;
    display: block;
    object-fit: contain; /* Esta línea está duplicada */
}

/* Después */
.icon {
    height: 8.9em;
    object-fit: contain;
    display: block;
}
```

b. Falta una familia de fuentes genérica

- **Descripción:** La declaración de font-family en global.css no incluye una familia de fuentes genérica, lo cual puede causar problemas de visualización si la fuente principal no carga.
- **Ubicación:** client/src/global.css (línea 7)
- **Solución:** Añadir una familia de fuentes genérica (ej. sans-serif) como respaldo.

```
/* Antes */
font-family: "Uncutsans";

/* Después */
font-family: "Uncutsans", Helvetica, Arial, Verdana, Tahoma, sans-serif;
```

c. Elementos no interactivos con click handlers sin keyboard listener

- **Descripción:** Elementos visibles y no interactivos que tienen un onClick handler deberían tener también un onKeyDown listener para mejorar la accesibilidad.
- **Ubicación:** client/src/components/ProjectHeader/ProjectHeader.jsx (línea 9)
- **Solución:** Añadir un onKeyDown listener al elemento.

```
{/* Antes */}
<div className={styles.menuContainer} onClick={() => setMenuOpen(!menuOpen)}>

{/* Después (ejemplo) */}
<div
    className={styles.menuContainer}
    onClick={() => setMenuOpen(!menuOpen)}
    onKeyDown={(e) => { /* manejar evento de teclado, ej. si e.key === 'Enter' */ }}
    role="button" // Se recomienda añadir un rol para accesibilidad
    tabIndex="0" // Y hacer el elemento enfocable
>
```

2. Security Hotspots (Puntos Calientes de Seguridad)

Los security hotspots son áreas del código que requieren una revisión manual para determinar si existe una vulnerabilidad de seguridad.

a. Imagen "node" con usuario root por defecto en Dockerfile

- **Descripción:** La imagen base node de Docker Hub se ejecuta con root como usuario por defecto, lo cual es un riesgo de seguridad en producción.
- **Ubicación:** server/Dockerfile (línea 1)
- **Solución:** Crear un usuario no-root dentro del Dockerfile y cambiar a él antes de ejecutar la aplicación.

b. Copia recursiva de datos sensibles al contenedor en Dockerfile

- **Descripción:** La instrucción COPY . . copia todo el contenido del directorio actual al contenedor, pudiendo incluir datos sensibles accidentalmente.
- **Ubicación:** server/Dockerfile (línea 9)

- **Solución 1 (Recomendada):** Utilizar un archivo .dockerignore para excluir archivos y directorios sensibles.

```
# Dockerfile
FROM node:20
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . . # Mantener esta línea, pero con .dockerignore

# .dockerignore (ejemplo)
.git
.env
node_modules
npm-debug.log
```

- **Solución 2:** Copiar solo los archivos y directorios específicos que la aplicación necesita (más laborioso de mantener).

```
# Ejemplo de solución 2
FROM node:20
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY src ./src
COPY public ./public
# ... copiar solo lo necesario
```

c. Uso de protocolo HTTP inseguro en Nodemailer

- **Descripción:** La configuración actual de Nodemailer podría llevar a una comunicación no cifrada o a una dependencia de un servidor SMTP que no fuerza TLS/SSL.
- **Ubicación:** server/src/utils/sendEmail.js (línea 4)
  - **Solución:** Configurar Nodemailer para forzar el uso de TLS/SSL (HTTPS).

const nodemailer = require("nodemailer");

```
const nodemailer = require("nodemailer");

let transporter = nodemailer.createTransport({
    service: "gmail", // O el host de tu SMTP
    auth: {
        user: process.env.GMAIL_USER,
        pass: process.env.GMAIL_PASS
    },
    secure: true, // Usa TLS/SSL
    requireTLS: true, // Fuerza TLS
    port: 465, // Puerto estándar para SMTPS (SMTP sobre SSL/TLS)
    // secured: true // Esta propiedad es redundante si secure: true
});
```

d. CORS habilitado sin configuración segura

- **Descripción:** La configuración actual de CORS es demasiado permisiva y podría permitir que sitios web maliciosos realicen solicitudes a tu API.
- **Ubicación:** server/src/index.js (línea 21)
- **Solución:** Configurar el middleware cors para permitir solicitudes solo desde orígenes (dominios) explícitamente confiables.

e. Omisión de --ignore-scripts en npm install

- **Descripción:** Omitir --ignore-scripts durante npm install en el Dockerfile puede permitir la ejecución de scripts maliciosos de dependencias.
- **Ubicación:** server/Dockerfile (línea 7)
- **Solución:** Añadir la bandera --ignore-scripts al comando npm install.

```
# Antes
RUN npm install

# Después
RUN npm install --ignore-scripts
```

f. Divulgación implícita de información de versión del framework

- **Descripción:** Express.js (o el framework usado) divulga implícitamente información de versión a través del encabezado HTTP X-Powered-By.
- **Ubicación:** server/src/index.js (línea 14, donde se inicializa express())
- **Solución:** Deshabilitar el encabezado X-Powered-By en Express.js.
  const app = express();

```
const app = express();
app.disable("x-powered-by"); // Añadir esta línea
```

3. Code Smells.

Un "Code Smell" es un síntoma en el código fuente que indica un problema de diseño o mantenibilidad. Aunque el código funciona, dificulta su comprensión y futuras modificaciones, aumentando la deuda técnica del proyecto.

En esta fase de desarrollo, nuestra prioridad es establecer una base funcional y segura, como se ha reflejado en la corrección de los bugs y los security hotspots. Muchos de los "Code Smells" detectados son comunes en las primeras etapas de un proyecto, donde la velocidad y la funcionalidad son clave. Por lo tanto, incluiremos las correcciones de code smells cuando la aplicación esté en una fase más madura.