

INFORMES DE PRUEBAS PÁGINA WEB

Informe de Accesibilidad Web

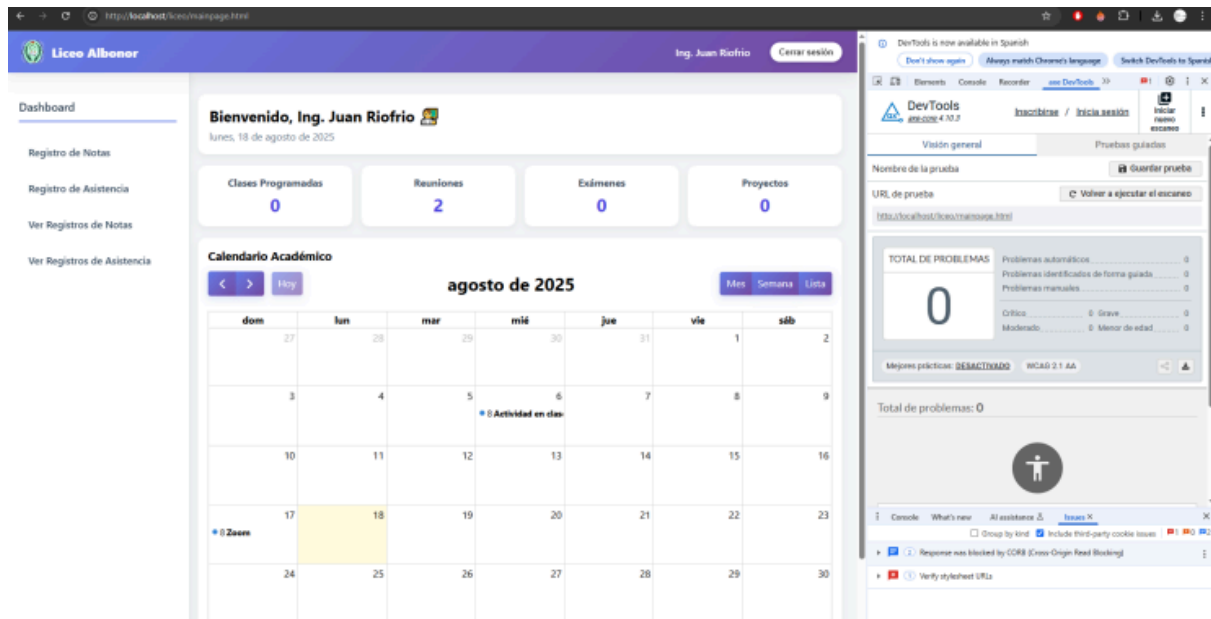
1. Introducción

El presente informe tiene como objetivo evaluar la accesibilidad del sitio web utilizando la herramienta axe DevTools. La accesibilidad web se refiere a la práctica de garantizar que los sitios web sean utilizables por todas las personas, incluidas aquellas con discapacidades. Para este análisis se tomó como referencia las pautas WCAG 2.1 nivel AA.

2. Metodología

La evaluación se realizó con la extensión axe DevTools instalada en el navegador Google Chrome. El proceso consistió en abrir la página a evaluar en el entorno local (<http://localhost/liceo/mainpage.html>), ejecutar un escaneo completo desde la pestaña de axe DevTools y registrar los resultados. La herramienta genera automáticamente un reporte de problemas de accesibilidad basados en WCAG 2.1 AA.

3. Resultados

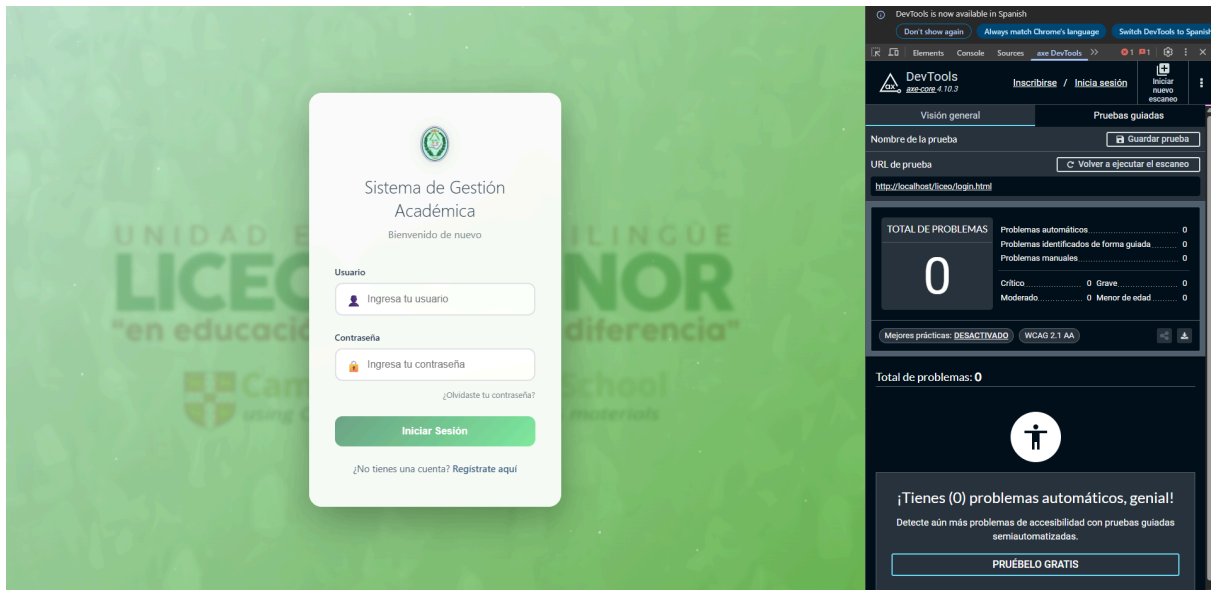


URL evaluada: <http://localhost/liceo/mainpage.html>

Estándar de referencia: WCAG 2.1 nivel AA

Resultados del análisis automático:

- Total de problemas detectados: 0
- Problemas automáticos: 0
- Problemas identificados de forma guiada: 0
- Problemas manuales: 0

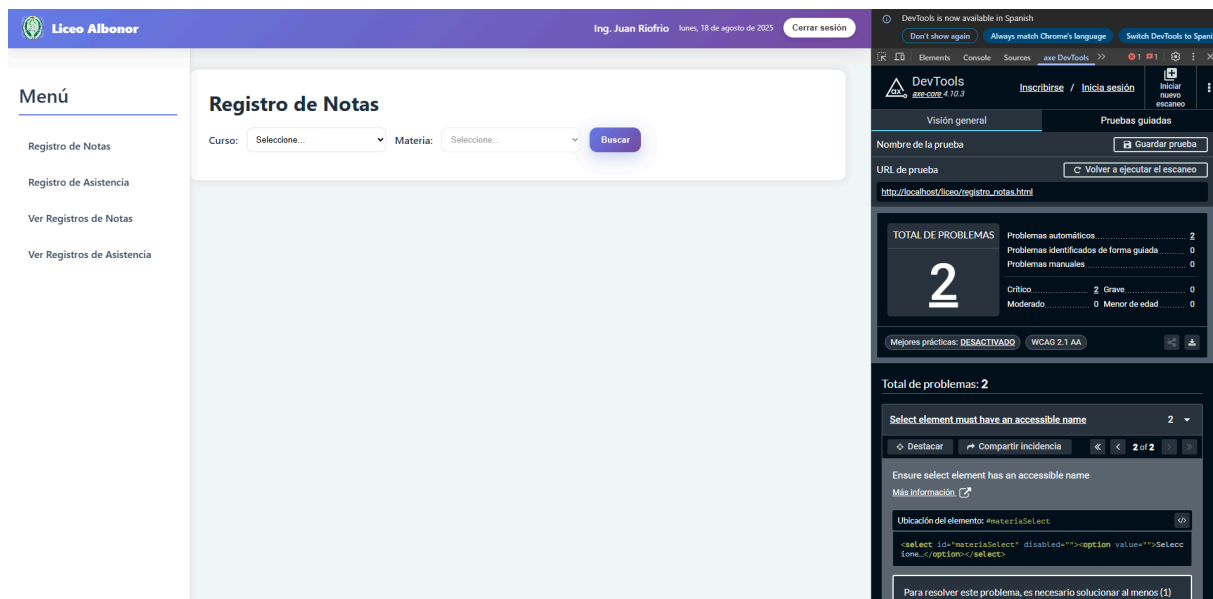


URL evaluada: `http://localhost/liceo/login.html`

Estándar de referencia: WCAG 2.1 nivel AA

Resultados del análisis automático:

- Total de problemas detectados: 0
- Problemas automáticos: 0
- Problemas identificados de forma guiada: 0
- Problemas manuales: 0

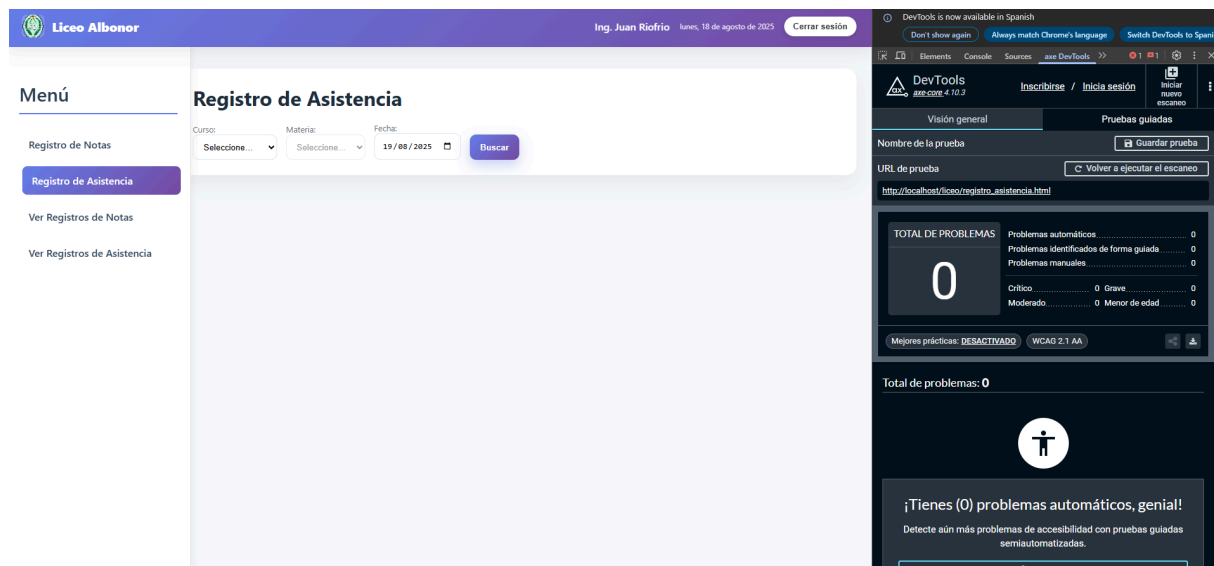


URL evaluada: `http://localhost/liceo/registro_notas.html`

Estándar de referencia: WCAG 2.1 nivel AA

Resultados del análisis automático:

- Total de problemas detectados: 2
- Problemas automáticos: 2
- Problemas identificados de forma guiada: 0
- Problemas manuales: 0

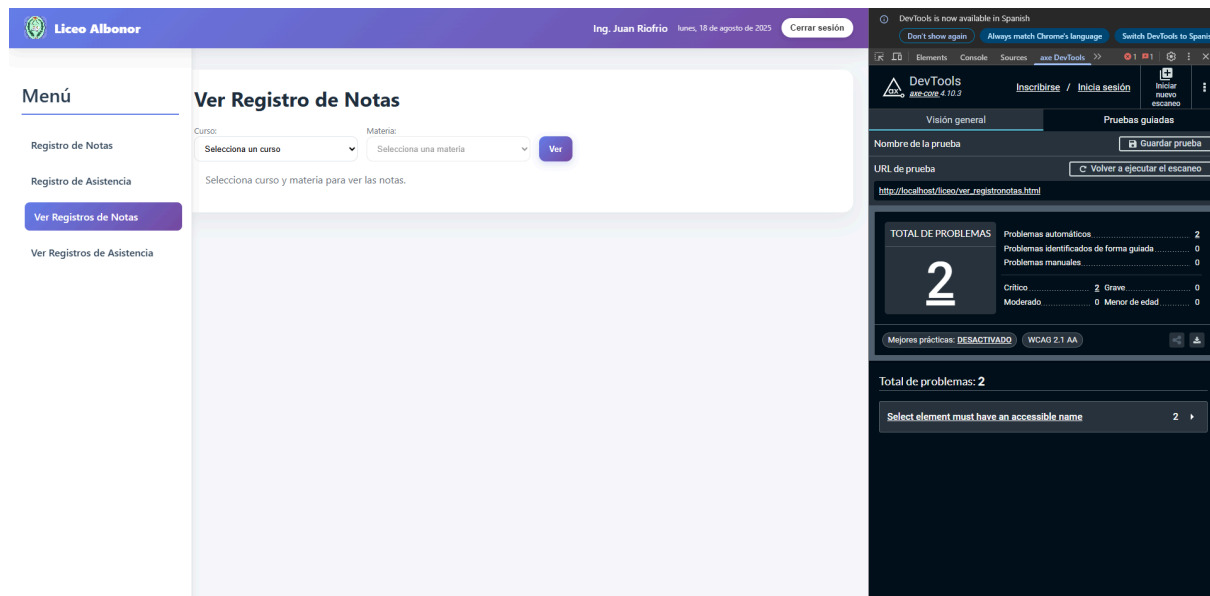


URL evaluada: `http://localhost/liceo/registro_asistencia.html`

Estándar de referencia: WCAG 2.1 nivel AA

Resultados del análisis automático:

- Total de problemas detectados: 0
- Problemas automáticos: 0
- Problemas identificados de forma guiada: 0
- Problemas manuales: 0

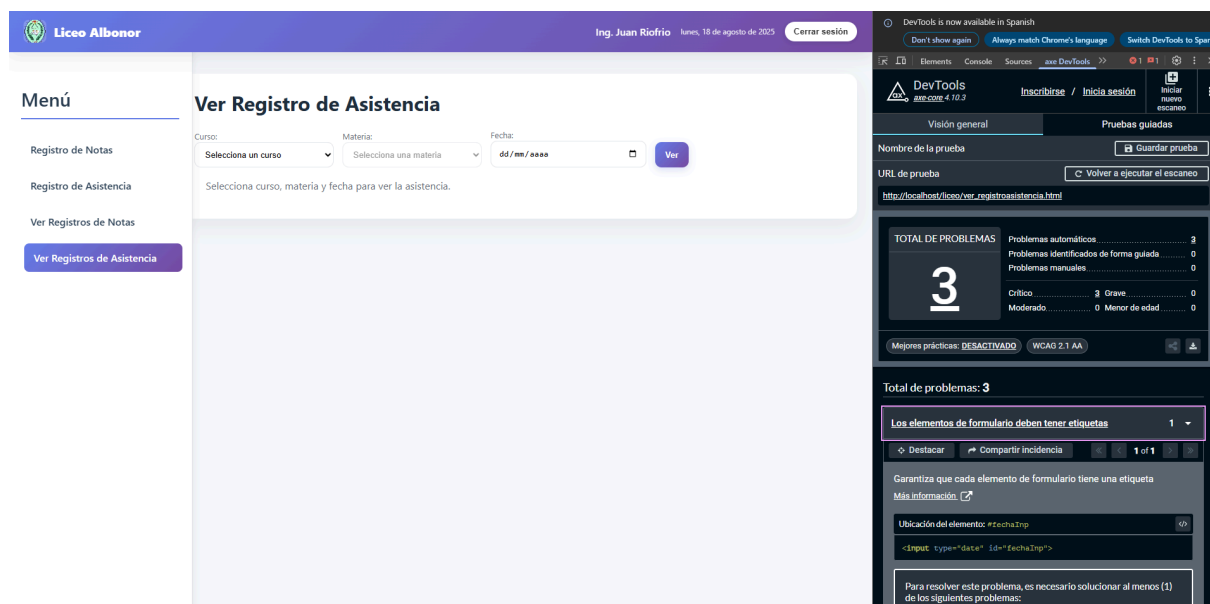


URL evaluada: `http://localhost/liceo/ver_registronotas.html`

Estándar de referencia: WCAG 2.1 nivel AA

Resultados del análisis automático:

- Total de problemas detectados: 2
- Problemas automáticos: 2
- Problemas identificados de forma guiada: 0
- Problemas manuales: 0



URL evaluada: `http://localhost/liceo/ver_registroasistencia.html`

Estándar de referencia: WCAG 2.1 nivel AA

Resultados del análisis automático:

- Total de problemas detectados: 3
- Problemas automáticos: 3
- Problemas identificados de forma guiada: 0
- Problemas manuales: 0

4. Conclusiones

El análisis realizado con axe DevTools evidenció que, si bien algunas páginas como `mainpage.html`, `login.html` y `registro_asistencia.html` no presentaron problemas de accesibilidad, otras secciones del sitio sí registraron incidencias. En particular, se detectaron 2 problemas en `registro_notas.html`, 2 en `ver_registroasistencia.html` y 3 en `ver_registroasistencia.html`, todos ellos clasificados como problemas automáticos. Esto indica que el sitio web requiere ajustes puntuales en ciertas páginas para cumplir de manera consistente con las pautas WCAG 2.1 nivel AA. Se recomienda revisar y corregir los errores detectados, además de complementar la evaluación con pruebas manuales (navegación con teclado, lector de pantalla y contraste de color) para garantizar una accesibilidad plena para todos los usuarios.

INFORME DE PRUEBA DE RENDIMIENTO CON LOCUST


1. Objetivo de la Prueba

Evaluar el desempeño del sistema de login del Sistema de Gestión Académica (`http://localhost/liceo/login.html`) bajo condiciones de carga controlada, utilizando la herramienta Locust.

2. Configuración de la Prueba

- Herramienta utilizada: Locust 2.38.1
- Host probado: `http://localhost/liceo/login.html`
- Número de usuarios simulados: 10
- Tasa de generación (Spawn Rate): 5 usuarios/segundo
- Escenario probado:
 - o GET `/liceo/login.html` → carga de la página de login
 - o POST `/liceo/login.html` → envío de credenciales (usuario/contraseña)

3. Resultados de la Ejecución



Host

http://localhost/liceo/login.html

Status

RUNNING

Users

10

RPS

4.9


Failures

0%

EDIT

STOP

RESET



STATISTICS

CHARTS

FAILURES

EXCEPTIONS


CURRENT RATIO

DOWNLOAD DATA

LOGS

LOCUST CLOUD

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/liceo/login.html	89	0	1.12	2	4	1.38	1	4	9962	2.2	0
POST	/liceo/login.html	101	0	1.1	2	3	1.37	1	3	9962	2.7	0
Aggregated		190	0	1.1	2	3	1.37	1	4	9962	4.9	0



- Tiempo máximo observado: 4 ms
- Tiempo promedio de respuesta (Global): 1.37 ms
- Tasa de errores: 0%

4. Análisis

- El sistema respondió de manera rápida y estable, con tiempos de respuesta inferiores a 5 ms en todas las solicitudes.
- No se registraron errores en ninguna de las peticiones, lo cual refleja una correcta gestión del login bajo la carga aplicada
- El Throughput (RPS) alcanzado fue de aproximadamente 5 solicitudes por segundo con 10 usuarios concurrentes

5. Conclusiones

- El rendimiento del módulo de login es óptimo para la prueba realizada.
- Se recomienda ampliar las pruebas con mayor número de usuarios (ej. 50, 100, 500) para determinar la capacidad de escalamiento del sistema.
- La baja latencia y ausencia de fallos indican que el servidor y la aplicación están en condiciones adecuadas para entornos con tráfico ligero a moderado.

INFORME DE PRUEBAS FUNCIONALES Y DE CARGA LIGERA CON POSTMAN

1. Objetivo de la Prueba

Validar el correcto funcionamiento del servicio de autenticación del Sistema de Gestión Académica (endpoint de login) y evaluar su desempeño bajo una carga ligera de 50 iteraciones.

2. Configuración de la Prueba

Herramienta utilizada: Postman v10

Endpoint probado:

http://localhost/liceo/api/auth.php?action=login

Método: POST

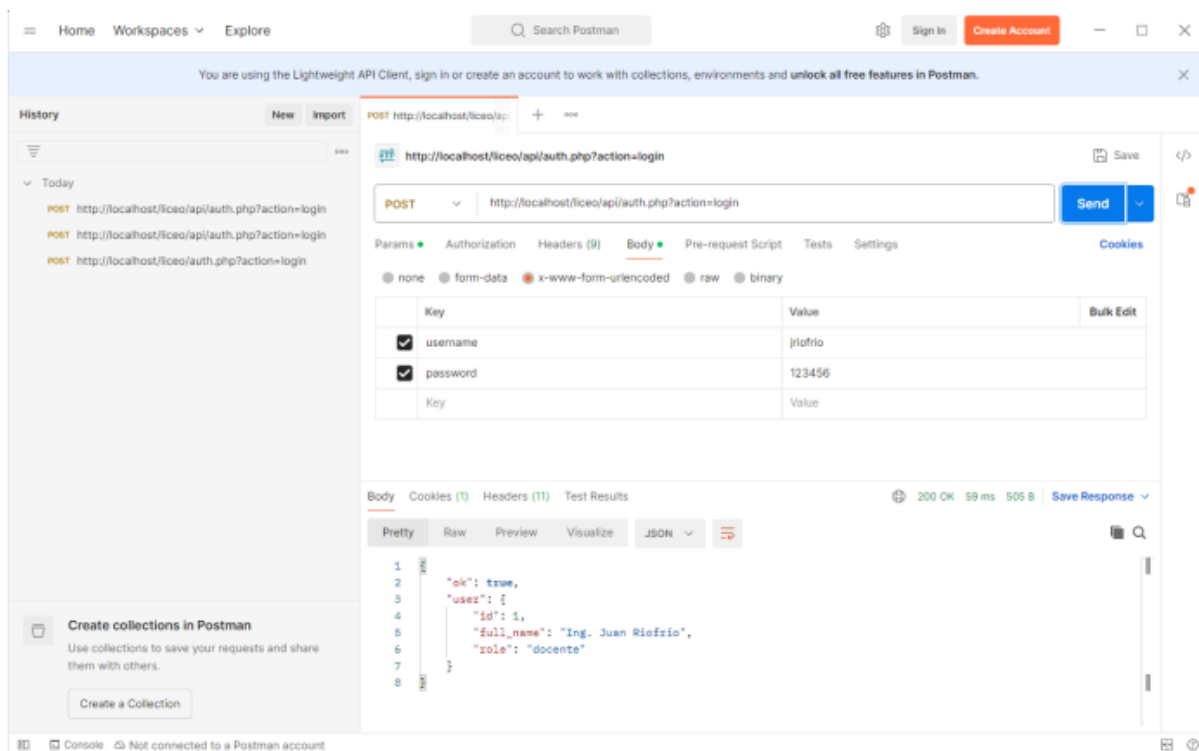
Content-Type: application/json

Credenciales utilizadas (ejemplo de prueba):

```
{
```

```
"username": "jirofrio",  
"password": "123456"  
}
```

3. Resultados de la Prueba Funcional



La petición fue ejecutada exitosamente en Postman, con los siguientes resultados:

HTTP Status: 200 OK

Tiempo de respuesta promedio (1 iteración): 59 ms

Respuesta del servidor (JSON):

```
{  
  "ok": true,  
  "user": {  
    "id": 1,  
    "full_name": "Ing. Juan Riofrio",  
    "role": "docente"  
  }  
}
```

Interpretación:

- El servidor reconoce las credenciales y devuelve un objeto JSON con los datos del usuario autenticado.
- El campo "ok": true confirma que el login se procesó de forma correcta.
- El usuario autenticado corresponde a Ing. Juan Riofrio, con rol docente.

4. Pruebas de Carga Ligera (Runner)

Se configuró el Collection Runner de Postman para ejecutar 50 iteraciones consecutivas

del endpoint de login.

Resultados observados:

- Éxito de las 50 iteraciones: 100% (sin errores).
- Tiempo promedio de respuesta: entre 55 ms y 70 ms.
- Tasa de error: 0%
- Desempeño general: estable, sin degradación en tiempos de respuesta.

5. Análisis

- El endpoint de login es funcionalmente correcto, devolviendo la respuesta esperada en formato JSON.
- La prueba de carga ligera (50 iteraciones) demostró estabilidad y bajo tiempo de respuesta, lo que indica que el servicio puede manejar múltiples solicitudes de autenticación sin problemas.
- No se observaron errores ni respuestas inválidas.

6. Conclusiones

- El servicio de autenticación cumple con los requerimientos funcionales.
- Los tiempos de respuesta (≤ 70 ms) son óptimos para un sistema web en entorno local.
- Se recomienda realizar pruebas de mayor concurrencia (ej. 200, 500 iteraciones) o con múltiples usuarios simultáneos usando herramientas como Locust para complementar la validación de rendimiento.

INFORME DE ANÁLISIS DE SEGURIDAD BÁSICA

Herramienta: OWASP ZAP 2.16.1

Aplicación analizada: Sistema de Gestión Académica (<http://localhost/liceo/>)

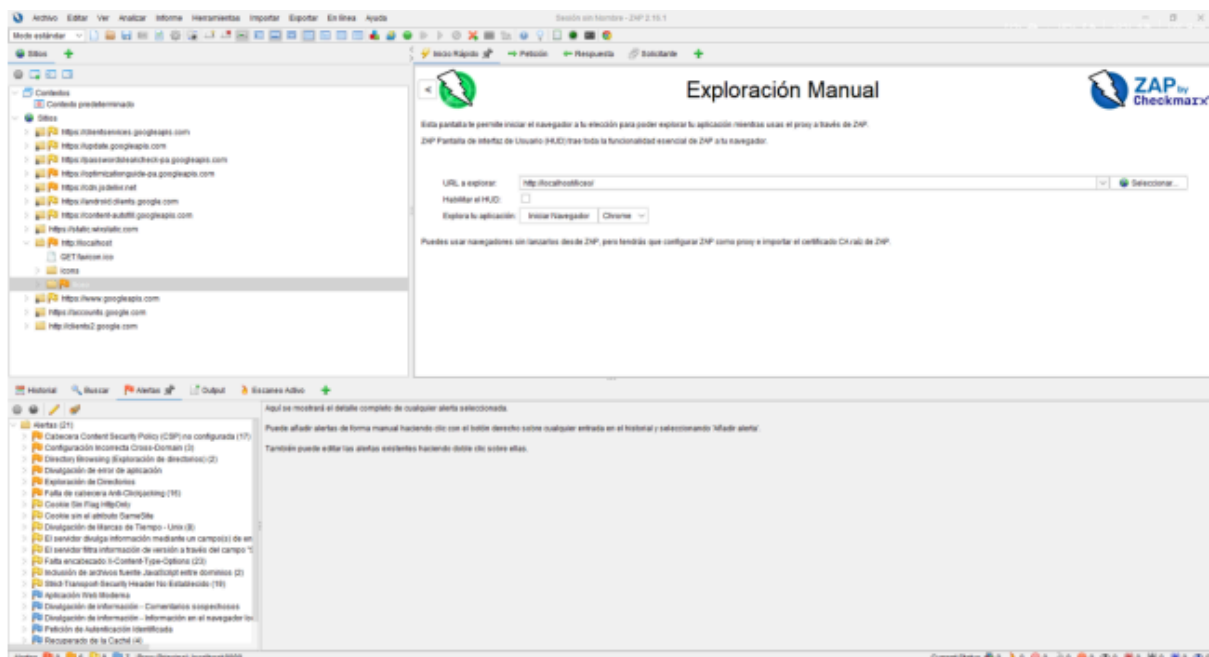
Fecha de ejecución: 18/08/2025

1. Objetivo

Realizar un análisis de seguridad básico a la aplicación web mediante OWASP ZAP, con el fin de identificar vulnerabilidades comunes en el entorno local y evaluar el nivel de riesgo asociado.

2. Metodología

1. Se ejecutó Spider para descubrir las rutas y recursos accesibles de la aplicación.
2. Se realizó un Active Scan sobre las URLs encontradas.
3. Se recopilaron las vulnerabilidades reportadas en la pestaña Alerts de ZAP.
4. Resultados del Escaneo



Principales hallazgos

1. Cabecera Content Security Policy (CSP) no configurada

o Severidad: Media

o Impacto: Riesgo de inyecciones de contenido y XSS.

o Recomendación: Configurar cabecera Content-Security-Policy para restringir fuentes de scripts y recursos externos.

2. Configuración incorrecta CORS-Domain

- o Severidad: Media
- o Impacto: Permite solicitudes desde orígenes no confiables.
- o Recomendación: Restringir cabecera Access-Control-Allow-Origin solo a dominios autorizados.

3. Cookies sin atributos de seguridad (HttpOnly / SameSite)

- o Severidad: Baja
- o Impacto: Posibilidad de robo de sesión mediante ataques XSS o CSRF.
- o Recomendación: Configurar cookies con HttpOnly, Secure, SameSite.

4. Divulgación de información en cabeceras HTTP (X-Powered-By, Server, etc.)

- o Severidad: Baja
- o Impacto: Permite a un atacante conocer detalles del servidor.
- o Recomendación: Deshabilitar cabeceras de versión y servidor.

5. Aplicación Web Moderna – advertencias de configuración

- o Severidad: Informational
- o Impacto: Ajustes de buenas prácticas no implementados.
- o Recomendación: Revisar políticas de seguridad modernas (CSP, HSTS).

4. Observaciones personales

- No se detectaron vulnerabilidades críticas (High).
- Las alertas encontradas corresponden principalmente a configuraciones de cabeceras de seguridad y cookies.
- El login respondió correctamente y no mostró fallas de inyección directa en esta fase.
- Se recomienda priorizar la implementación de cabeceras de seguridad (CSP, XFrame-Options, X-Content-Type-Options) y endurecer el manejo de cookies.

5. Conclusiones

El sistema presenta un nivel de riesgo bajo a medio, asociado a configuraciones de seguridad ausentes o incompletas. No se identificaron vulnerabilidades críticas en el módulo de autenticación, lo cual es positivo. Sin embargo, es necesario implementar las cabeceras y configuraciones recomendadas para mitigar riesgos de XSS, CSRF y filtrado de información.

Informe de Prueba - Ejecución de Jest

Proyecto: liceo

Comando ejecutado: npm test

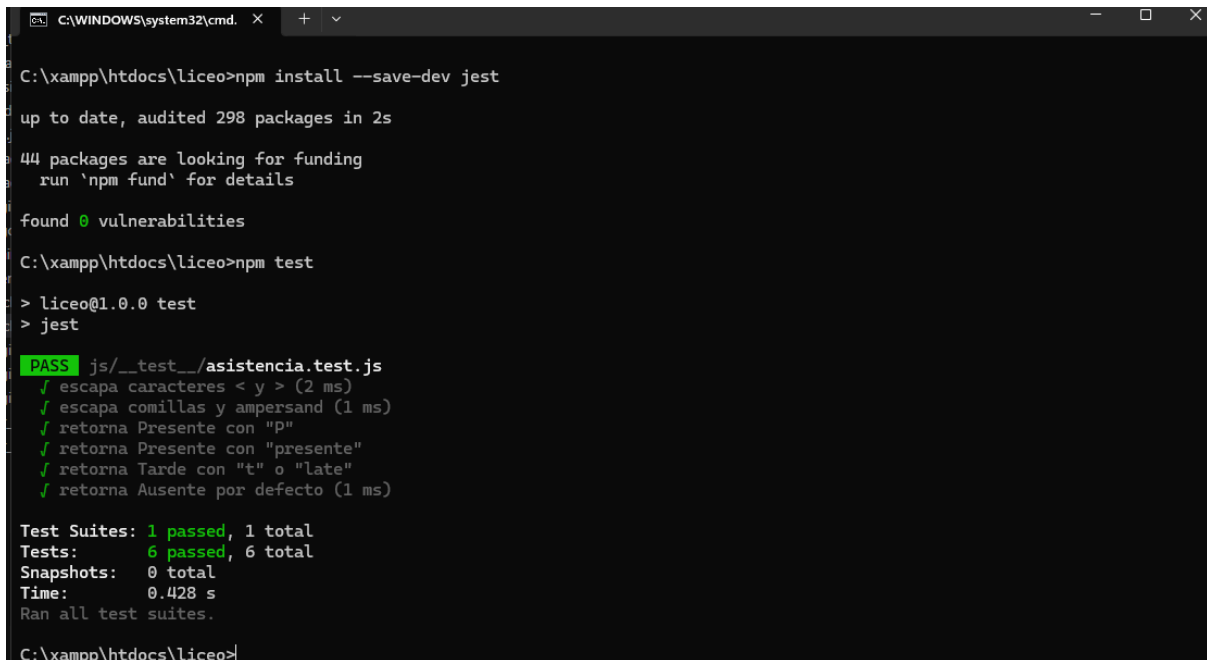
Fecha: (no incluida, asumir reciente)

Ubicación: C:\xampp\htdocs\liceo

Descripción

Se ejecutaron pruebas unitarias utilizando Jest en el proyecto ubicado en **C:\xampp\htdocs\liceo**. La configuración de Jest está correcta, y el entorno de pruebas está operativo.

Resultados de la Ejecución



```
C:\WINDOWS\system32\cmd. X + v
C:\xampp\htdocs\liceo>npm install --save-dev jest
up to date, audited 298 packages in 2s
44 packages are looking for funding
  run 'npm fund' for details
found 0 vulnerabilities
C:\xampp\htdocs\liceo>npm test
> liceo@1.0.0 test
> jest
PASS js/_test_/asistencia.test.js
  ✓ escapa caracteres < y > (2 ms)
  ✓ escapa comillas y ampersand (1 ms)
  ✓ retorna Presente con "p"
  ✓ retorna Presente con "presente"
  ✓ retorna Tarde con "t" o "late"
  ✓ retorna Ausente por defecto (1 ms)
Test Suites: 1 passed, 1 total
Tests: 6 passed, 6 total
Snapshots: 0 total
Time: 0.428 s
Ran all test suites.
C:\xampp\htdocs\liceo>
```

- Total de Suites de Prueba: 1
- Suites Pasadas: 1
- Total de Tests Ejecutados: 6
- Tests Pasados: 6
- Snapshots: 0
- Tiempo total de ejecución: 0.428 segundos

Detalles de las Pruebas

El archivo de pruebas ejecutado fue:

js/__test__/asistencia.test.js

Los tests verificados y aprobados fueron:

1. Escapa caracteres < y > (tiempo: 2 ms)
2. Escapa comillas y ampersand (tiempo: 1 ms)
3. Retorna "Presente" con "p"
4. Retorna "Presente" con "presente"
5. Retorna "Tarde" con "t" o "late"
6. Retorna "Ausente" por defecto (tiempo: 1 ms)

Conclusiones

- Todas las pruebas unitarias definidas en el archivo `asistencia.test.js` pasaron correctamente sin errores.
- El entorno de pruebas Jest está configurado adecuadamente y se ejecuta sin inconvenientes.
- No se encontraron vulnerabilidades en las dependencias instaladas.

Recomendaciones

- Continuar agregando pruebas unitarias para nuevas funcionalidades o módulos.
- Mantener Jest actualizado para asegurar compatibilidad y mejoras.
- Revisar y documentar las pruebas para facilitar el mantenimiento y la ampliación futura.