Plan de Pruebas y Ciberseguridad DevSecOps

1. Introducción

El objetivo de este plan de pruebas es garantizar la calidad, seguridad y desempeño de la aplicación mediante una estrategia de pruebas que abarque desde pruebas unitarias hasta pruebas de seguridad avanzadas, integradas en el pipeline de CI/CD. Se aplicará el concepto de *Shift Left Testing* para detectar y corregir defectos lo antes posible, incluyendo pruebas de ciberseguridad para detectar vulnerabilidades comunes (inyección SQL, CSRF, XSS, etc.) y realizar pentesting.

2. Alcance y Objetivos

Alcance

- Gestión de evaluaciones (registro, redirección y visualización en historial).
- Eliminación segura de registros mediante validación CSRF.
- Validación de entradas y protección contra inyecciones.
- Verificación de tokens CSRF en formularios.
- Pruebas de pentesting básicas (inyección SQL, XSS).
- Medición de tiempos de respuesta y estabilidad bajo carga.
- Automatización de pruebas en cada commit (CI/CD).
- Pruebas de contenido E2E, navegación y base de datos mediante ORM.

Objetivos

- Confirmar que los endpoints y flujos de usuario se comportan como se espera.
- Detectar vulnerabilidades con análisis SAST, DAST, IAST y pentesting.
- Evaluar el rendimiento en escenarios de carga y estrés.
- Integrar todas las pruebas en pipelines automatizados y mantener seguridad continua.

3. Estrategia de Pruebas

3.1 Tipos de Pruebas

A. Funcionales

- **Pruebas Unitarias:** Validación de funciones específicas como `determinarCategoriaRiesgo`, validadores y helpers.
- **Pruebas de Integración:** Validación de comunicación entre módulos (controladores, rutas, base de datos).
- **Pruebas E2E (Cypress):** Simulación completa del flujo del usuario, desde la evaluación hasta el historial.
- **Pruebas de Navegación (Playwright):** Validación del recorrido visual e interacción entre vistas del sistema.

B. No Funcionales

- **Pruebas de Rendimiento:** Simulación de carga en endpoints como `/Historial` con usuarios virtuales concurrentes.
- **Pruebas de Concurrencia:** Ejecución de miles de solicitudes concurrentes usando `k6`.
- **Pruebas de Usabilidad:** Evaluaciones manuales para mejorar la experiencia de usuario.

C. Pruebas de Seguridad

- **SAST (Static Application Security Testing):** Análisis de código con ESLint y (pendiente) SonarQube.
- **DAST (Dynamic Application Security Testing):** Pruebas activas con OWASP ZAP y Burp Suite.
- **Pentesting básico:** Pruebas de inyección SQL, XSS y CSRF manuales y automatizadas.
- **IAST / RASP:** (Opcional) Herramientas para monitoreo en tiempo real y prevención activa.

D. Pruebas de Base de Datos (ORM)

- Validación de integridad, inserciones, actualizaciones y eliminaciones con Sequelize, facilitando pruebas limpias y aisladas.

3.2 Integración en CI/CD

- Uso de GitHub Actions para pipelines automatizados por cada `push`.
- Ejecución continua de pruebas unitarias, integración, E2E, navegación y rendimiento.
- Automatización de escáneres de seguridad DAST y alertas en caso de fallos.
- Monitoreo activo mediante despliegue tipo canary.

4. Herramientas y Recursos

Tipo de Prueba	Herramienta Sugerida
Pruebas Unitarias	Jest, Mocha
Pruebas de Integración	Supertest, Postman
Pruebas E2E / Contenido	Cypress
Pruebas de Navegación	Playwright
Análisis SAST	ESLint (SonarQube pendiente)
Análisis DAST / Pentesting	OWASP ZAP, Burp Suite
Pruebas de Carga / Concurrencia	k6
CI/CD	GitHub Actions
ORM	Sequelize
Monitoreo	ELK Stack, Datadog

5. Entorno de Pruebas

- **Local:** Node.js, Express, MySQL con entorno de pruebas controlado.
- **CI/CD:** GitHub Actions con ejecución automática de pruebas.
- **Staging:** Entorno previo a producción para pruebas finales y escaneos de seguridad.

6. Ejecución y Cronograma

1. Configuración Inicial y Seeding.
2. Desarrollo de pruebas unitarias, de integración, E2E y navegación.
3. Configuración del pipeline en GitHub Actions para ejecución automática.
4. Validaciones finales en entorno staging.
5. Despliegue gradual con monitoreo en producción.

7. Roles y Responsabilidades

- **Desarrollador/Tester:**
    - Diseñar y mantener las pruebas.
    - Automatizar y configurar el CI/CD.
    - Ejecutar pruebas de seguridad y rendimiento.
    - Documentar resultados y dar seguimiento a los errores detectados.

8. Métricas y KPIs

- **Cobertura de pruebas:** > 80% en código.
- **Tiempo de respuesta:** < 300 ms en promedio.
- **Vulnerabilidades críticas:** 0 en escaneos SAST/DAST.
- **Estabilidad bajo carga:** Disponibilidad > 99.9%.
- **Errores en BD o concurrencia:** < 1% en pruebas de estrés.

9. Reporte y Seguimiento de Defectos

- Uso de GitHub Issues o herramientas similares para documentar y dar seguimiento.
- Retroalimentación iterativa para ajustar estrategias.
- Notificaciones automáticas en caso de fallos en pipelines.