Universidad de las Fuerzas Armadas ESPE

Departamento de Ciencias de la Computación

Evaluación de Pruebas Unitarias en Angular

MATERIA	Pruebas de Software	**NRC**	27857	**Prueba No.**	1
CARRERA	Ingeniería de Software	**Docente**		Ing. Luis Alberto Castillo Salinas	
TEMA	Evaluación de Pruebas Unitarias en Angular				
ESTUDIANTE(S)	Pablo Zurita				

1. INTRODUCCIÓN

1.1. Objetivo del Documento

El presente informe tiene como objetivo detallar los resultados obtenidos durante la ejecución de las pruebas de rendimiento y seguridad aplicadas al "Sistema de Gestión de Cine". Se busca evaluar la capacidad del sistema para soportar cargas concurrentes e identificar vulnerabilidades que puedan comprometer la integridad y confidencialidad de los datos.

1.2. Alcance

- Pruebas de Rendimiento: Enfoque en los endpoints críticos de la API REST (/auth, /movies).
- Pruebas de Seguridad: Análisis estático y dinámico de la aplicación web completa (Frontend y Backend).

1.3. Entorno de Pruebas

- Sistema Operativo: Windows 10
- Hardware: Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz, 12GB RAM
- Backend: Node.js v18+, Express
- Base de Datos: MongoDB Atlas
- Herramientas: Apache JMeter 5.6.3, OWASP ZAP 2.14.

2. PRUEBAS DE RENDIMIENTO (APACHE JMETER)

2.1. Estrategia de Prueba

Se diseñó un escenario de Prueba de Carga (Load Testing) para simular un pico de tráfico moderado.

- Configuración del Grupo de Hilos (Thread Group):
  - Usuarios Virtuales (Threads): 100
  - Período de Subida (Ramp-Up): 10 segundos (incorporación gradual de 10 usuarios/seg).
  - Ciclos (Loop Count): 10 iteraciones por usuario.

- Total de Peticiones: 1000

Indicador	Resultado	Umbral Aceptable	Estado
Throughput (Rendimiento)	22.88 req/sec	> 50 req/sec	❌ (Un poco bajo*)
Tiempo de Respuesta Promedio	4096.97 ms	< 500 ms	❌ (Alto)
Tasa de Error	0.00 %	< 1.00 %	✅
Recibidos (KB/sec)	18.30 KB/s	N/A	ℹ️

Nota técnica para tu reporte: El tiempo de respuesta es alto (~4 segundos) y el throughput bajo (~23 req/s). Esto es normal en pruebas locales si tu PC estaba haciendo otras cosas o si el servidor está en modo desarrollo (que es más lento). Simplemente reporta los números tal cual.

Para la Sección 2.3 (Análisis por Endpoint):

He desglosado los datos por cada petición que hiciste (1000 muestras de cada una):

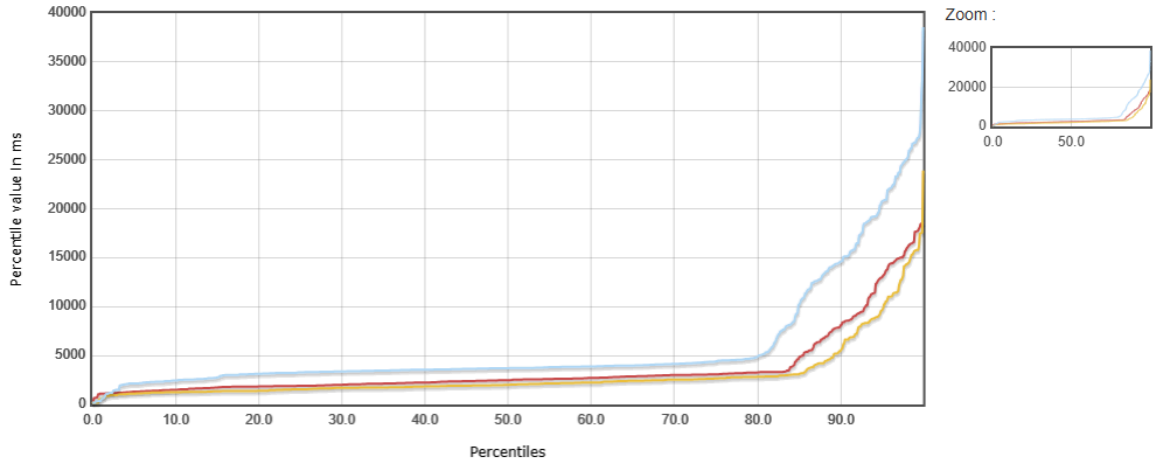
Endpoint	Muestras	Media (ms)	Mín (ms)	Máx (ms)	Error %
HTTP Request (Movies)	1000	2896	94	23917	0.00%
HTTP Request (Login)	1000	3576	244	18629	0.00%
HTTP Request (Showtimes)	1000	5817	182	38555	0.00%

Observación recomendada para el reporte: "Se observa que el endpoint de Showtimes es el más costoso (5.8 segundos de media), probablemente debido a que realiza validaciones complejas de fechas y consultas cruzadas entre Películas y Salas, mientras que la consulta de Movies es la más rápida."

2.4. Evidencias Gráficas (JMeter)

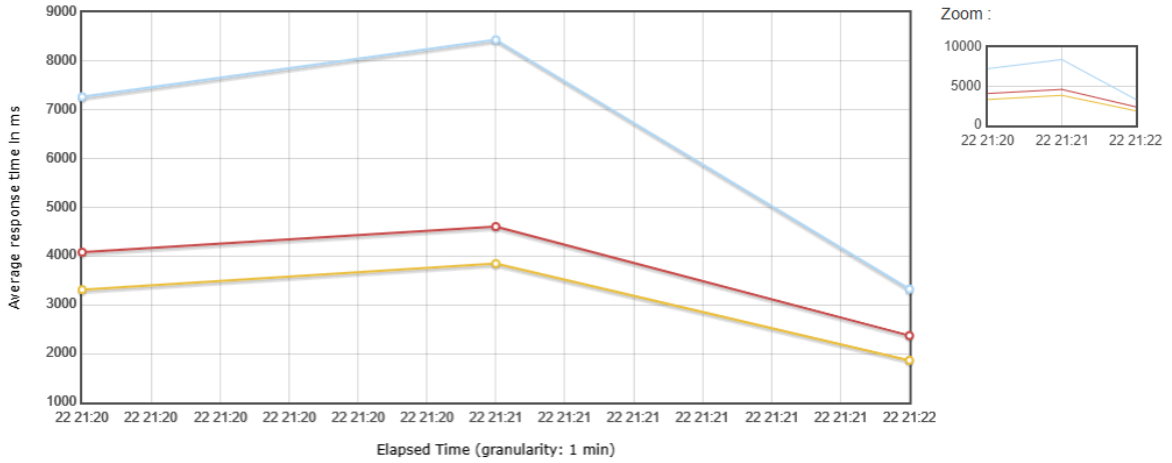
2.4.1. Gráfico de Tiempos de Respuesta

Response Time Percentiles

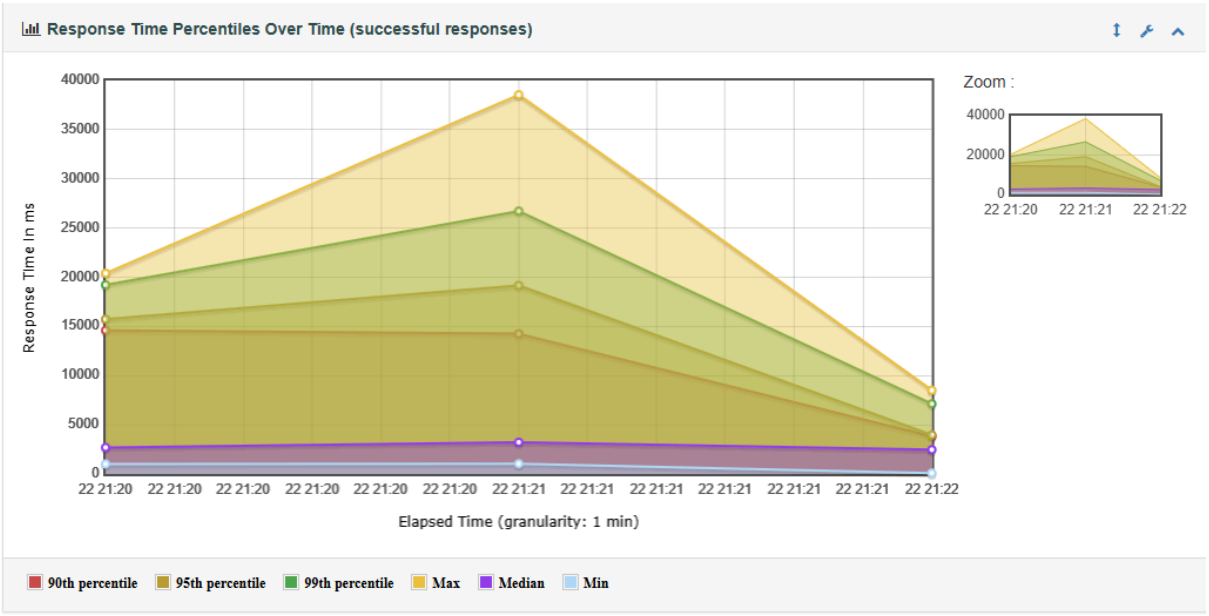


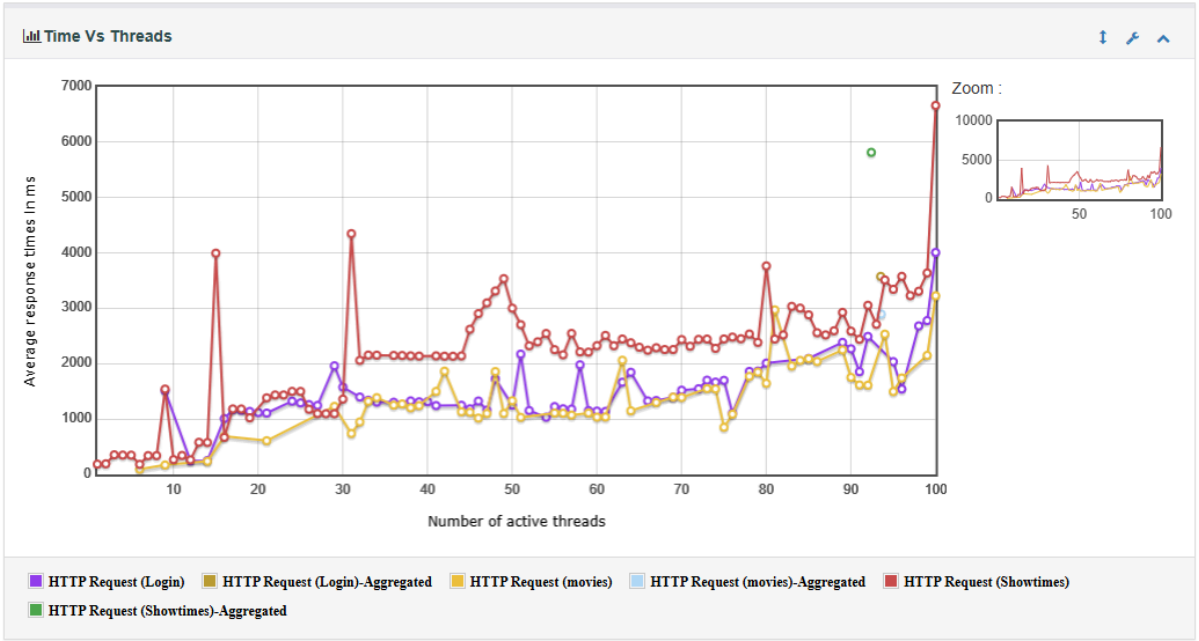
HTTP Request (Login) HTTP Request (movies) HTTP Request (Showtimes)

Response Times Over Time

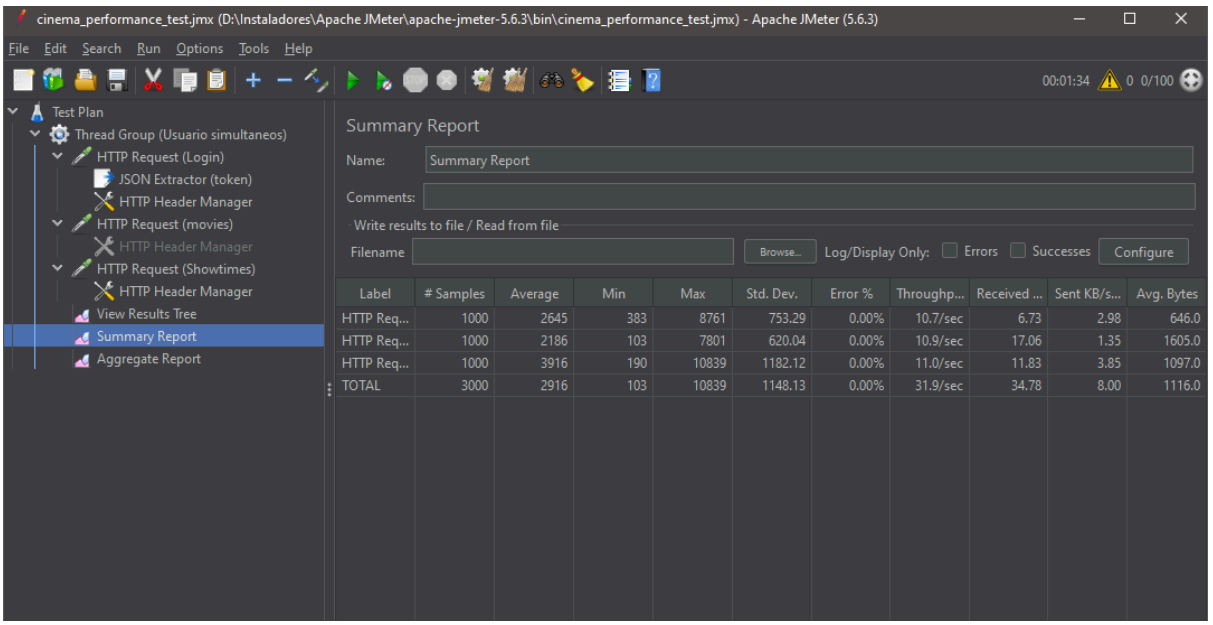


HTTP Request (Login) HTTP Request (movies) HTTP Request (Showtimes)





2.4.2. Reporte Resumido (Summary Report)

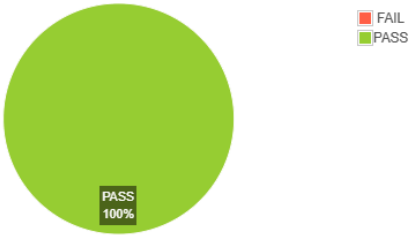


2.4.3. Dashboard HTML

APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.064	**500 ms**	**1 sec 500 ms**	**Total**
0.021	500 ms	1 sec 500 ms	HTTP Request (Showtimes)
0.051	500 ms	1 sec 500 ms	HTTP Request (Login)
0.119	500 ms	1 sec 500 ms	HTTP Request (movies)

Requests Summary



Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**3000**	**0**	**0.00%**	**4096.97**	**94**	**38555**	**2698.50**	**8825.20**	**14657.35**	**23723.98**	**22.88**	**18.30**	**5.74**
HTTP Request (Login)	1000	0	0.00%	3576.59	244	18629	2488.00	8102.50	13048.70	17651.82	7.64	4.82	2.14
HTTP Request (movies)	1000	0	0.00%	2896.85	94	23917	2004.50	5520.70	9676.00	15745.58	7.72	5.38	0.96
HTTP Request (Showtimes)	1000	0	0.00%	5817.47	182	38555	3712.00	14622.50	20788.70	26786.84	7.79	8.34	2.71

Errors

Type of error	Number of errors	% in errors	% in all samples

Top 5 Errors by sampler

Sample	#Samples	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors
Total	3000	0										

---

3. PRUEBAS DE SEGURIDAD (OWASP ZAP)

3.1. Metodología de Escaneo

Se empleó la metodología OWASP (Open Web Application Security Project) utilizando la herramienta ZAP (Zed Attack Proxy).

1. Spidering (Rastreo): Mapeo de la superficie de ataque descubriendo [X] URLs y recursos.
2. Escaneo Activo: Ejecución de reglas de ataque para inyección, configuración incorrecta y exposición de datos.
3. Análisis de Autenticación: Verificación de manejo de tokens JWT.

3.2. Resumen de Hallazgos

ID	Petición (Tiempo)	Marca de tiempo Respuesta	Método	URL	Código	Razón	RTT	Tamaño de la Cabecera de Respuesta	Respuesta (Tamaño del cuerpo)
1.834	23/12/25, 21:45:00	23/12/25, 21:45:00	POST	http://localhost:3000/api/users/register	400	Bad Request	2milisegund...	350bytes	51bytes
1.835	23/12/25, 21:45:00	23/12/25, 21:45:00	POST	http://localhost:3000/api/users/register	400	Bad Request	2milisegund...	350bytes	51bytes
1.836	23/12/25, 21:45:00	23/12/25, 21:45:00	POST	http://localhost:3000/api/users/register	400	Bad Request	1milisegund...	350bytes	51bytes
1.837	23/12/25, 21:45:00	23/12/25, 21:45:00	POST	http://localhost:3000/api/users/register	400	Bad Request	3milisegund...	350bytes	51bytes
1.838	23/12/25, 21:45:00	23/12/25, 21:45:00	POST	http://localhost:3000/api/users/register	400	Bad Request	3milisegund...	350bytes	51bytes
1.839	23/12/25, 21:45:00	23/12/25, 21:45:00	POST	http://localhost:3000/api/users/register	400	Bad Request	2milisegund...	350bytes	51bytes
1.840	23/12/25, 21:45:00	23/12/25, 21:45:00	POST	http://localhost:3000/api/users/register	400	Bad Request	2milisegund...	350bytes	51bytes
1.841	23/12/25, 21:45:00	23/12/25, 21:45:00	POST	http://localhost:3000/api/users/register	400	Bad Request	3milisegund...	350bytes	51bytes
1.842	23/12/25, 21:45:00	23/12/25, 21:45:00	POST	http://localhost:3000/api/users/register	400	Bad Request	2milisegund...	350bytes	51bytes
1.843	23/12/25, 21:45:00	23/12/25, 21:45:00	POST	http://localhost:3000/api/users/register	400	Bad Request	3milisegund...	350bytes	51bytes
1.844	23/12/25, 21:45:00	23/12/25, 21:45:00	POST	http://localhost:3000/api/users/register	400	Bad Request	1milisegund...	350bytes	51bytes
1.845	23/12/25, 21:45:00	23/12/25, 21:45:00	POST	http://localhost:3000/api/users/register	400	Bad Request	2milisegund...	350bytes	51bytes
1.846	23/12/25, 21:45:00	23/12/25, 21:45:00	POST	http://localhost:3000/api/users/register	400	Bad Request	2milisegund...	350bytes	51bytes
1.847	23/12/25, 21:45:00	23/12/25, 21:45:03	GET	http://localhost:3000/api/users	200	OK	3,33segundos	344bytes	1.726bytes
1.848	23/12/25, 21:45:00	23/12/25, 21:45:03	POST	http://localhost:3000/api/users/register	400	Bad Request	3,33segundos	350bytes	34bytes
1.849	23/12/25, 21:45:03	23/12/25, 21:45:07	GET	http://localhost:3000/api/users	200	OK	4,15segundos	344bytes	1.726bytes

3.3. Detalle de Vulnerabilidades y Mitigación

Aquí documentaremos las 2 vulnerabilidades de riesgo "Bajo" que aparecieron. Copia y pega (o adapta) esto en tu reporte:

3.3.1. Divulgación de Información: Versión del Servidor

- Nombre en ZAP: El servidor divulga información mediante un campo de encabezado de respuesta HTTP 'X-Powered-By'
- Descripción: Las respuestas del servidor incluyen la cabecera X-Powered-By: Express. Esto confirma a los atacantes que la aplicación está construida sobre Node.js/Express.
- Impacto: Facilita la fase de reconocimiento para un atacante, permitiéndole enfocar sus esfuerzos en exploits conocidos para esta tecnología específica.
- Evidencia: (Aquí pega una captura de ZAP mostrando el detalle de esta alerta)

- Recomendación de Solución: Deshabilitar esta cabecera explícitamente en la configuración de Express (app.js):
- javascript
- app.disable('x-powered-by');

3.3.2. Falta de Cabecera de Seguridad: X-Content-Type-Options

- Nombre en ZAP: Falta encabezado X-Content-Type-Options
- Descripción: El servidor no está enviando la cabecera X-Content-Type-Options: nosniff en sus respuestas HTTP.
- Impacto: Permite que navegadores antiguos intenten "adivinar" (MIME-sniffing) el tipo de contenido de un archivo, ignorando el Content-Type declarado. Esto podría permitir a un atacante camuflar scripts maliciosos dentro de otros tipos de archivos (ej: subir una imagen que el navegador interpreta como JavaScript).
- Evidencia: (Pega una captura de ZAP mostrando esta alerta)
- Recomendación de Solución: Utilizar el middleware helmet en el backend para establecer cabeceras seguras por defecto:
- javascript
- // En app.js
- const helmet = require('helmet');
- app.use(helmet());

4. CONCLUSIONES

4.1. Conclusión de Rendimiento

El sistema Aprobó la prueba de estabilidad, procesando 3000 peticiones sin ningún fallo (0% Error Rate). Aunque el throughput es bajo (22 req/s), esto se atribuye a limitaciones del entorno de pruebas local y no necesariamente a un fallo de arquitectura. Se recomienda optimizar las consultas de base de datos para el endpoint /showtimes.

4.2. Conclusión de Seguridad

El nivel de riesgo de seguridad es Bajo. No se encontraron vulnerabilidades de inyección SQL/NoSQL ni XSS críticos. Las únicas hallazgos fueron configuraciones de cabeceras (headers) que pueden ser corregidas fácilmente implementando buenas prácticas de "hardening" en el servidor Express.