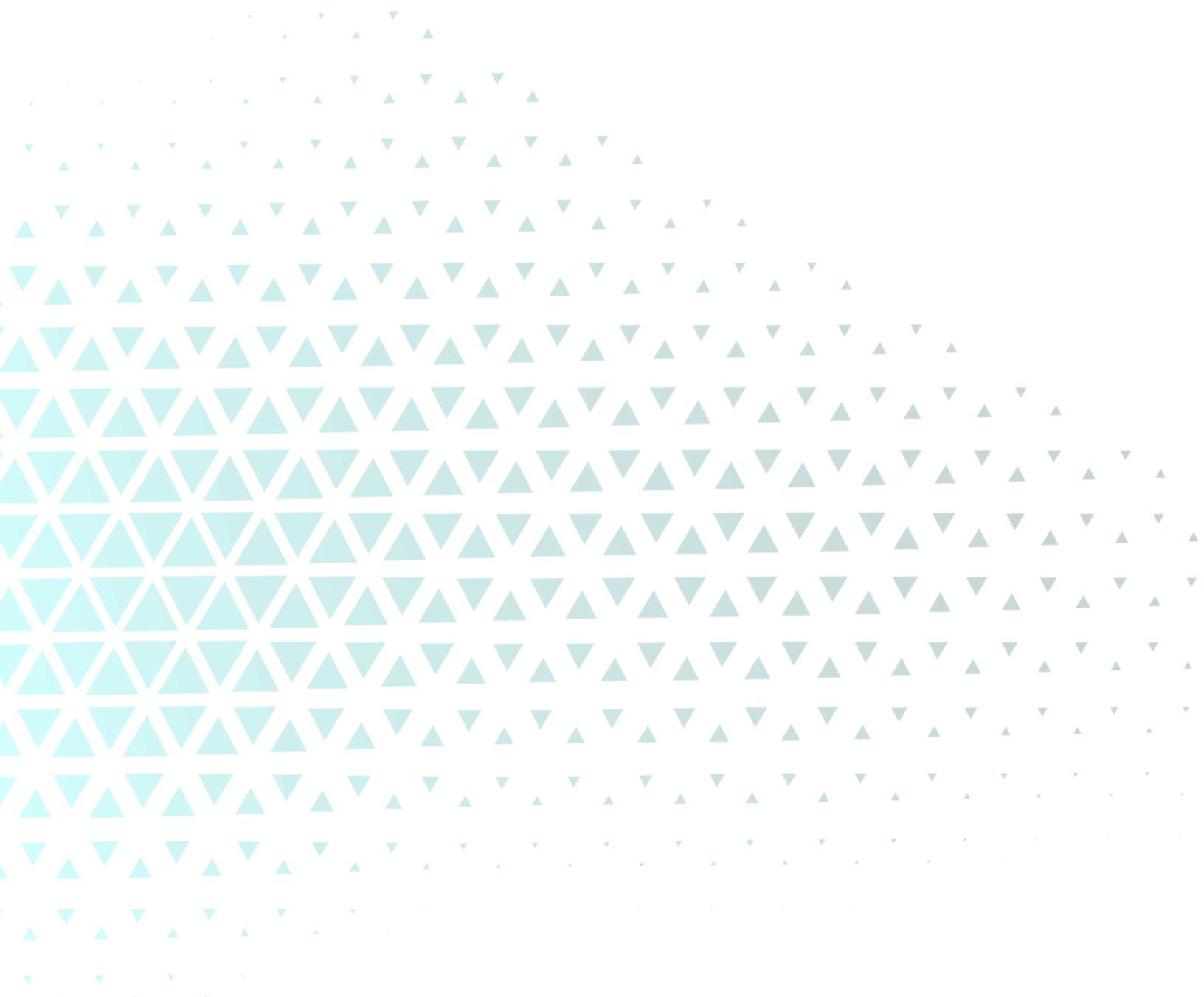


# Informe de Pentest



## Índice

1. Introducción
2. Alcance y objetivos
3. Metodología
4. Resultados
5. Vulnerabilidades
6. Recomendaciones
7. Conclusiones

<b>Versión</b>	<b>Fecha</b>	<b>Auditores</b>	<b>Cambios</b>
1.0	18/12/2025	Ignacio Muñoz González	Primera versión
1.0	18/12/2025	Daniel Barón Iglesias	Primera versión
1.0	18/12/2025	David Prieto González-Hidalgo	Primera versión
1.0	18/12/2025	Roberto Benítez Martín	Primera versión
1.0	18/12/2025	Jon Ormaechea Caro	Primera versión
1.0	18/12/2025	Arturo Valverde Carrasco	Primera versión
1.0	18/12/2025	Alejandro Alcázar Lucas	Primera versión
1.0	18/12/2025	Alberto Seco Fuente	Primera versión

## 1. Introducción

Este informe presenta los resultados del análisis de seguridad realizado sobre una aplicación corporativa que integra una plataforma web, un chatbot basado en modelos de lenguaje (LLM) y una base de datos con información sensible de empleados, clientes, productos y ventas. El objetivo principal del pentest es evaluar la robustez de las defensas del sistema, identificar vulnerabilidades técnicas y lógicas, y determinar el riesgo asociado a la interacción entre el chatbot, la capa de generación de consultas y los datos internos de la organización.

La aplicación permite a los usuarios autenticados realizar consultas en lenguaje natural, que son interpretadas por un modelo de IA y traducidas a operaciones sobre la base de datos. Este diseño introduce una superficie de ataque particular, donde fallos en la validación de entradas, en la generación de consultas o en la gestión de permisos pueden derivar en accesos no autorizados, filtración de datos o correlaciones indebidas entre información de RRHH y marketing/ventas.

El presente documento recoge el alcance, la metodología aplicada y los hallazgos obtenidos, con el fin de proporcionar a los equipos de desarrollo, datos y seguridad una referencia clara para la mejora continua del sistema.

## 2. Alcance y objetivos

### Alcance

El alcance de esta auditoría abarca el análisis completo de la aplicación, incluyendo el chatbot basado en IA, la capa que genera consultas a partir del lenguaje natural, los endpoints expuestos y la base de datos que contiene información de empleados, clientes, productos y ventas.

La evaluación se realizó en un entorno controlado y autorizado, limitándose estrictamente a los componentes y funcionalidades definidos en el alcance, sin efectuar pruebas sobre sistemas externos o no contemplados. El análisis se orientó a evaluar la seguridad del sistema desde la perspectiva de un atacante, reproduciendo escenarios realistas para identificar fallos que pudieran permitir accesos indebidos, correlaciones no autorizadas o

exposiciones de datos que afectaran a la confidencialidad, integridad o disponibilidad de la información corporativa

## Objetivos

Los objetivos principales de este test de penetración son los siguientes:

- Identificar vulnerabilidades técnicas y lógicas derivadas del flujo conversacional y de la generación dinámica de consultas.
- Evaluar el impacto potencial de accesos indebidos a datos sensibles, correlaciones entre tablas y exposición indirecta de información.
- Verificar la existencia de controles adecuados de autenticación, autorización y segmentación de permisos.
- Analizar la superficie de ataque introducida por el LLM, incluyendo riesgos de prompt injection, overfetching y model leakage.
- Documentar las pruebas realizadas, las herramientas empleadas y las evidencias obtenidas.
- Proponer mejoras que permitan asegurar la aplicación y garantizar su viabilidad en un entorno corporativo.

## Propósito del Informe

El propósito final de este informe es servir como documento de referencia para los distintos departamentos involucrados, facilitando la implementación de las acciones correctivas necesarias. De este modo, se busca garantizar que la aplicación alcance un nivel adecuado de seguridad, estabilidad y funcionalidad, alineado con las buenas prácticas y estándares de la industria.

### 3. Metodología

La metodología aplicada sigue el estándar PTES (Penetration Testing Execution Standard), adaptado a las particularidades de una aplicación que combina IA conversacional, generación dinámica de consultas y acceso a datos sensibles. Las fases ejecutadas fueron las siguientes:

#### 1. Reglas de compromiso

Se definió como alcance la aplicación completa en un entorno controlado, incluyendo chatbot, backend, endpoints y base de datos. Se acordó realizar pruebas no destructivas, orientadas a la identificación y explotación controlada de vulnerabilidades.

#### 2. Recolección de información

Se recopiló información sobre la arquitectura del sistema, los flujos entre chatbot, capa de consultas y base de datos, la estructura de tablas y sus relaciones, los roles y permisos definidos en USERS, los endpoints y sus mecanismos de autenticación, así como el comportamiento del LLM ante distintos tipos de prompts. Todo ello se contrastó con los equipos de desarrollo, datos y backend para asegurar una visión precisa del funcionamiento real de la aplicación.

#### 3. Modelado de Amenazas y priorización

A partir de la información recopilada se identificaron como activos críticos los datos personales, el historial de ventas, la información de empleados y los logs del sistema, lo que permitió priorizar vectores de ataque relevantes como la prompt injection, la generación de consultas demasiado amplias, el acceso indirecto mediante correlaciones entre tablas, la exposición de estructura interna a través del chatbot y la existencia de permisos excesivos en la base de datos.

#### **4. Análisis de vulnerabilidades**

Se llevó a cabo un análisis manual y asistido por herramientas para identificar fallos en la validación de entradas, la generación de consultas SQL, la gestión de roles y privilegios, la seguridad en tránsito y almacenamiento de datos, así como en la configuración de los endpoints y APIs expuestos.

#### **5. Explotación y Escalado de Privilegios (Post-Explotación Inicial)**

Se verificaron de forma controlada las vulnerabilidades detectadas, evaluando el acceso a datos fuera del alcance del rol asignado, la obtención de información sensible mediante prompts manipulados, la posibilidad de correlaciones indebidas entre distintas tablas y la exposición de metadatos o detalles internos del sistema a través del chatbot.

#### **6. Post-Explotación**

Se analizaron las rutas potenciales que podrían permitir el acceso a datos sensibles adicionales, la obtención de mecanismos de persistencia dentro del sistema y el posible abuso de logs o información residual que pudiera revelar detalles internos o facilitar movimientos laterales.

#### **7. Documentación y Recomendaciones**

Se registraron evidencias, comandos, prompts utilizados, respuestas del sistema y análisis de impacto, junto con recomendaciones específicas para cada hallazgo.

## 4. Resultados

Verificamos el certificado del dominio para confirmar que es **segura, auténtica y confiable**.

Visor de certificados: onrender.com

**General** Detalles

Enviado a

Nombre común (CN)	onrender.com
Organización (O)	<No incluido en el certificado>
Unidad organizativa (OU)	<No incluido en el certificado>

Emitido por

Nombre común (CN)	WE1
Organización (O)	Google Trust Services
Unidad organizativa (OU)	<No incluido en el certificado>

Período de validez

Emitido el	domingo, 30 de noviembre de 2025, 20:53:50
Vencimiento el	sábado, 28 de febrero de 2026, 21:53:48

Huellas digitales SHA-256

Certificado	c2bb420fde9d27f30b28f59ed737438150bb04f2aa89413a285fef18069260b8
Clave pública	d278e5f668177f3ab78f0ec5a13ced4ce0da14fc4c7de0e3a06ee262d70472bc

El certificado ha sido emitido por **Google Trust Services (CN: WE1)** y está vigente desde el 30 de noviembre de 2025 **hasta el 28 de febrero de 2026**. También se incluyen las huellas digitales **SHA-256** tanto del certificado como de la clave pública, que permiten verificar su integridad y autenticidad. En conjunto, el dominio utiliza un certificado TLS válido y firmado por una autoridad de confianza.

## Enumeración pasiva (Shodan y Web-Check):

// TAGS: cdn

**General Information**

- Country: United States
- City: San Francisco
- Organization: Render
- ISP: Render
- ASN: AS397273

**Web Technologies**

- CDN: Cloudflare
- Miscellaneous: HTTP/3

**Open Ports**

80 / TCP

```

HTTP/1.1 301 Moved Permanently
Date: Wed, 17 Dec 2025 11:18:59 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 65
Connection: keep-alive
CF-RAY: 9af69d5d9ac27e865-SJC
Location: https://survey.columbiasc.gov/
cf-cache-status: DYNAMIC
Server: cloudflare
alt-svc: h3=":443"; ma=6400
  
```

184387524 | 2025-12-17T11:18:59.465791

Se consultó la dirección IP del frontend utilizando Shodan y Web-Check con el fin de recopilar información pública disponible, incluyendo servicios expuestos, puertos abiertos y datos asociados a la infraestructura.

15 jobs successful 4 jobs skipped 14 jobs failed

Show Details

Finished in 10800 ms

Dismiss

**SSL Certificate**

Subject: onrender.com  
Issuer: Google Trust Services prime256v1  
ASN1 Curve: P-256  
MIST Curve: P-256  
Expires: 28 February 2026  
Renewed: 30 November 2025  
Serial Num: D7ED90A458F37F113D...  
Fingerprint: A8:EE:46:11:10:0C:...

**Extended Key Usage**

TLS Web Server Authentication

**Social Tags**

Title: Desafio G1

**Threats**

Phishing Status: No Phishing Found

**Domain Whois**

Present

Having a security.txt ensures security researchers know how and where to safely report vulnerabilities

**Security.Txt**

Present

**Headers**

date: 17 December 2025  
content-type: 1 August 2001  
content-length: 258  
connection: keep-alive  
cf-ray: 9af7617789365678-LHR  
accept-ranges: bytes  
access-control-allow-credentials: true  
access-control-allow-origin: http://localhost:5173  
access-control-expose-headers: X-New-Access-Token  
cache-control: 1 January 2000  
cross-origin-opener-policy: same-origin  
cross-origin-resource-policy: same-origin  
etag: W/"1c3-19b2cadb85f"  
last-modified: 17 December 2025  
origin-agent-cluster: 1 January 2001  
referrer-policy: no-referrer  
rnd-id: 615a3651-42f1-4697  
strict-transport-security: max-age=1518600; includeSubDomains  
vary: Origin, Accept-Encoding  
x-content-type-options: nosniff  
x-dns-prefetch-control: off  
x-download-options: noopen  
x-frame-options: SAMEORIGIN  
x-permitted-cross-domain-policies: none  
x-render-origin-server: Render  
x-xss-protection: 1 January 2000  
cf-cache-status: DYNAMIC

**DNS Records**

A: 216.24.57.7  
AAAA: 216.24.57.251  
SOA: gcp-us-west1-1.origin.onrender.com

**HTTP Security**

Content Security Policy: No  
Strict Transport Policy: Yes  
X-Content-Type-Options: Yes  
X-Frame-Options: Yes  
X-XSS-Protection: Yes

**HSTS Check**

HSTS Enabled? No  
HSTS header does not contain the preload directive.

**Block Lists**

AddGuard: Not Blocked  
AdGuard Family: Not Blocked  
CleanBrowsing Adult: Not Blocked  
CleanBrowsing Family: Not Blocked  
CloudFlare: Not Blocked  
CloudFlare Family: Not Blocked  
Comodo Secure: Not Blocked  
Google DNS: Not Blocked  
Neustar Family: Not Blocked  
Neustar Protection: Not Blocked  
Norton Family: Not Blocked  
OpenDNS: Not Blocked  
OpenDNS Family: Not Blocked  
Quad9: Not Blocked

**TXT Records**



Desafio ABORTED

Summary Correlations Browse Graph Scan Settings Log

Search...

Type	Unique Data Elements	Total Data Elements	Last Data Element
Similar Domain	24	24	2025-12-17 15:06:05
Non-Standard HTTP Header	22	46	2025-12-17 11:24:14
Linked URL - Internal	13	17	2025-12-17 11:24:09
HTTP Headers	8	8	2025-12-17 11:24:11
Raw DNS Records	6	6	2025-12-17 11:23:41
Web Content	5	8	2025-12-17 11:24:11
HTTP Status Code	4	9	2025-12-17 11:24:11
Linked URL - External	4	4	2025-12-17 11:23:52

## Enumeración OSINT mediante SpiderFoot:

Se llevó a cabo un escaneo utilizando SpiderFoot con el objetivo de recopilar información pública (OSINT) asociada al objetivo, como dominios, direcciones IP, servicios expuestos, tecnologías utilizadas y posibles relaciones con terceros. El proceso se basó principalmente en fuentes públicas externas, sin interacción directa con los sistemas objetivo.

localhost:8834/#/scans/reports/44/hosts

**Desafío**

Back to My Scans

Hosts 2 Vulnerabilities 5 Notes 1 History 1

Filter Search Hosts 2 Hosts

Host	Auth	Vulnerabilities
216.24.57.251	Fail	37
216.24.57.7	Fail	21

Scan Details

Policy: Web Application Tests  
Status: Completed  
Severity Base: CVSS v3.0  
Scanner: Local Scanner  
Start: Today at 1:43 PM  
End: Today at 3:26 PM  
Elapsed: 2 hours

Vulnerabilities



Critical  
High  
Medium  
Low  
Info

Tenable News

Securing Federal Cloud Environments: Overcoming 5 ...

Read More

localhost:8834/#/scans/folders/my-scans

localhost:8834/#/scans/reports/44/vulnerabilities

**Desafío**

Back to My Scans

Hosts 2 Vulnerabilities 5 Notes 1 History 1

Filter Search Vulnerabilities 5 Vulnerabilities

Sev	CVSS	VPR	EPSS	Name	Family	Count	...
INFO	...	...	...	HTTP (Multiple Issues)	Web Servers	34	...
INFO				Nessus SYN scanner	Port scanners	20	...
INFO				Nessus Scan Information	Settings	2	...
INFO				Web Application Sitemap	Web Servers	1	...
INFO				Web Server No 404 Error Code Check	Web Servers	1	...

Scan Details

Policy: Web Application Tests  
Status: Completed  
Severity Base: CVSS v3.0  
Scanner: Local Scanner  
Start: Today at 1:43 PM  
End: Today at 3:26 PM  
Elapsed: 2 hours

Vulnerabilities



Critical  
High  
Medium  
Low  
Info

Tenable News

Dell Storage Manager Multiple Vulnerabilities

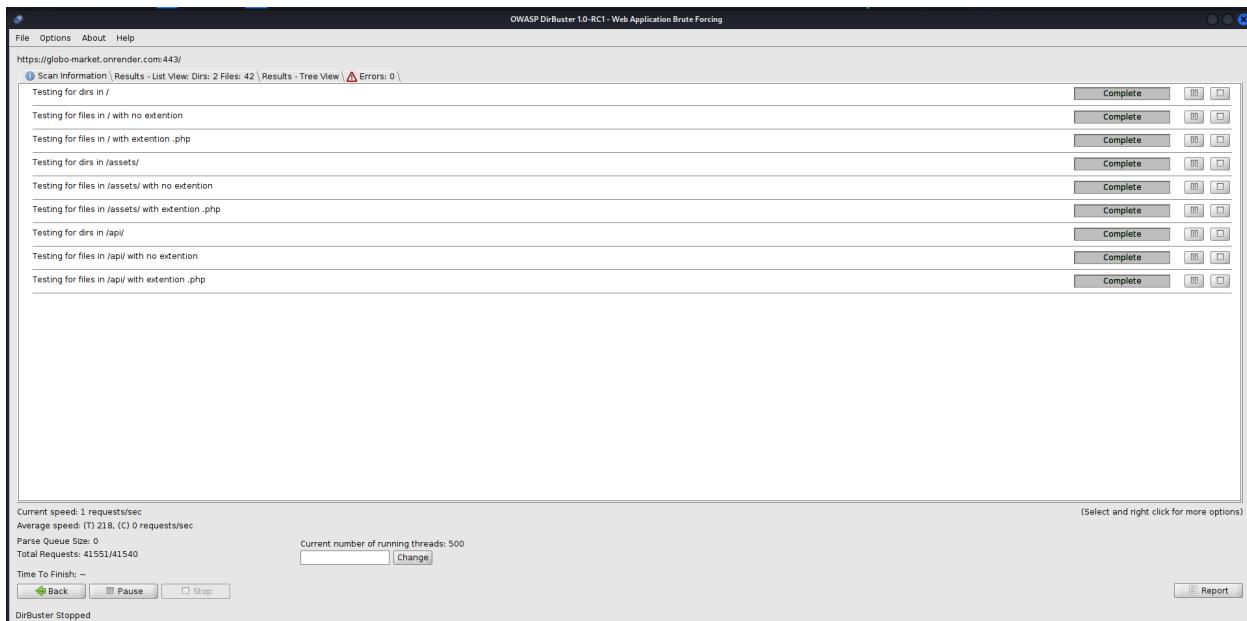
Read More

## Escaneo de vulnerabilidades web (Nessus):

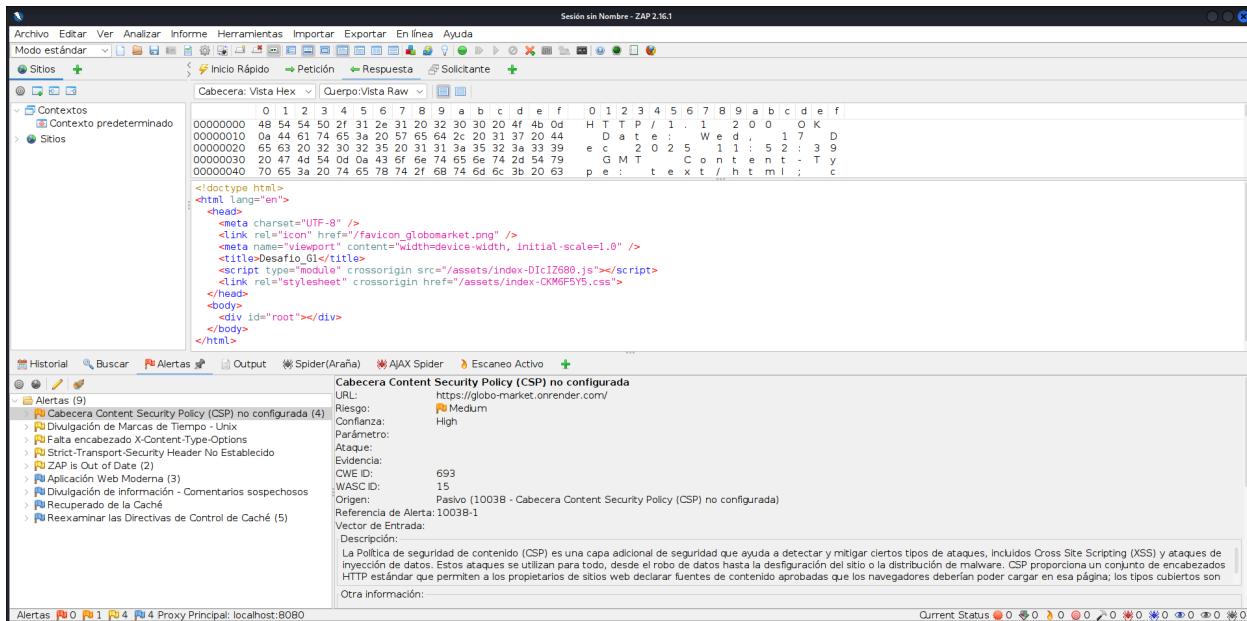
Se realizó un escaneo de vulnerabilidades web sobre el frontend utilizando Nessus, con el objetivo de identificar posibles debilidades de seguridad a nivel de configuración, servicios y componentes expuestos. Como parte de la entrega, se adjunta el reporte técnico generado por la herramienta con el detalle de los hallazgos detectados.

## Enumeración web (DirBuster):

Se llevó a cabo un proceso de enumeración de recursos web sobre el frontend mediante DirBuster, con el fin de identificar directorios y ficheros accesibles. Como resultado, se adjunta un pequeño reporte en formato `.txt` con los recursos descubiertos durante el escaneo.



### Análisis de vulnerabilidades web (OWASP ZAP):



Se realizó un análisis de seguridad del frontend utilizando la herramienta OWASP ZAP para identificar vulnerabilidades a nivel de aplicación web. Como resultado de este proceso, se detallan a continuación las vulnerabilidades detectadas durante el escaneo.

Desde Kali Linux realizamos una búsqueda mediante la herramienta Nmap sobre la dirección IP de la API, con el objetivo de identificar los puertos abiertos, los servicios en ejecución y posibles vectores de ataque que permitan evaluar su nivel de exposición y seguridad.

Mediante el siguiente comando:

```
nmap -sV -O 129.213.23.117
```

- **-sV**: Muestra la versión de los puertos abiertos
- **-O**: Muestra la posible versión de sistema operativo.

```
[root@kali] ~
# nmap -sV -O 129.213.23.117
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-18 12:34 CET
Nmap scan report for 129.213.23.117
Host is up (0.043s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
22/tcp    open  tcpwrapped
80/tcp    open  tcpwrapped
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: bridge/VoIP adapter/general purpose
Running (JUST GUESSING): Oracle Virtualbox (98%), Slirp (98%), AT&T embedded (94%), QEMU (93%)
OS CPE: cpe:/o:oracle:virtualbox cpe:/a:danny_gasparovski:slirp cpe:/a:qemu:qemu
Aggressive OS guesses: Oracle Virtualbox Slirp NAT bridge (98%), AT&T BGW210 voice gateway (94%), QEMU user mode network gateway (93%)
No exact OS matches for host (test conditions non-ideal).

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 20.70 seconds
```

Los resultados obtenidos indican que los puertos 22 y 80 se encuentran abiertos. No obstante, no es posible identificar las versiones de los servicios asociados, ya que ambos aparecen como tcpwrapped, lo que sugiere la presencia de mecanismos de control de acceso o filtrado que limitan la enumeración directa de los servicios.

- Utilizamos el comando **curl** para obtener las cabeceras **HTTP** del servidor, lo que ayuda a identificar su configuración y nivel de exposición.

```
[root@kali] /home/jo
# curl -I http://129.213.23.117/
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Thu, 18 Dec 2025 12:50:14 GMT
Content-Type: application/json
Content-Length: 50
Connection: keep-alive
Access-Control-Allow-Origin: http://localhost:5173
Access-Control-Allow-Credentials: true
Vary: Origin
```

```
curl -I http://129.213.23.117/
```

- **-I**: sirve para realizar una solicitud HEAD y obtener las cabeceras de la respuesta.

Los resultados indican que el servidor usa **nginx 1.18.0** sobre Ubuntu y expone esta información, lo que supone una filtración de metadatos.

También se observa que devuelve **JSON** por defecto y aplica una política **CORS** limitada a **localhost:5173**, propia de un entorno de desarrollo. Además, el uso de Access-Control-Allow-Credentials: true podría ser riesgoso si el origen no estuviera restringido. En conjunto, **las cabeceras muestran una configuración funcional pero que requiere endurecimiento antes de un entorno real.**

- Desde Kali Linux ejecutamos la herramienta **slowhttptest**, se utiliza para identificar configuraciones inseguras en servidores web, evaluar su capacidad para cerrar conexiones lentas, detectar posibles vulnerabilidades frente a ataques de denegación de servicio de baja tasa (Low-and-Slow) y validar la eficacia de los timeouts, los límites de conexión y otros mecanismos de mitigación implementados en el sistema.

Para ello utilizamos el comando:

```
slowhttptest -c 1000 -H -g -o slowhttp -i 10 -r 10 -t GET -u http://129.213.23.117/ -x 240 -p 3
```

- **-c 1000**: 1000 conexiones simultáneas
- **-H**: modo Slow Headers
- **-i 10**: intervalo de 10 segundos entre fragmentos de encabezado
- **-r 10**: 10 conexiones por segundo
- **-t GET**: método GET
- **-u**: URL objetivo
- **-x 240**: duración total de 240 segundos
- **-p 3**: tamaño del fragmento de encabezado

```

Thu Dec 18 11:32:28 2025:
    slowhttptest version 1.9.0
  - https://github.com/shekyan/slowhttptest -
test type:                      SLOW HEADERS
number of connections:          1000
URL:                            http://129.213.23.117/
verb:                           GET
cookie:
Content-Length header value:   4096
follow up data max size:       68
interval between follow up data: 10 seconds
connections per seconds:       10
probe connection timeout:      5 seconds
test duration:                 240 seconds
using proxy:                   no proxy

Thu Dec 18 11:32:28 2025:
slow HTTP test status on 160th second:

initializing:        0
pending:             0
connected:           40
error:               0
closed:              960
service available: YES
Thu Dec 18 11:32:32 2025:
Test ended on 164th second
Exit status: No open connections left

```

El resultado de la prueba Slow Headers muestra que, aunque el servidor aceptó inicialmente cientos de conexiones lentas, fue cerrándolas progresivamente hasta terminar con todas antes de los 240 segundos previstos, **manteniéndose disponible en todo momento**; esto indica que **cuenta con mecanismos de defensa** (timeouts o límites de conexión) **que evitan la denegación de servicio**, aunque sigue siendo susceptible a variantes más agresivas o distribuidas del ataque.

### Intento de fuerza bruta con Burpsuite:

Se capturó una petición de login de la página <https://globo-market.onrender.com/api>. Usando un diccionario creado para esta página, se intentó realizar una fuerza bruta con el **intruder**. Al intentar de hacer más de 5 intentos, el servidor dejó de responder a los intentos y se paró la fuerza bruta.

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		400	580		1109		
1	Adm1n123	401	497		985		
2	Adm1n@0	401	281		985		
3	Adm1naa1	401	281		985		
4	Adm1naa2	401	288		985		
5	Adm1naa3		0				

## Intento de fuerza bruta con XSS

Durante la auditoría realizada, se comprobó que los vectores de entrada habituales (formularios, parámetros en la URL y cabeceras) cuentan con mecanismos de validación y sanitización adecuados, lo que impide la ejecución de scripts externos. Por ello, se concluye que la aplicación auditada no es vulnerable a XSS en el estado actual, ya que las pruebas realizadas no lograron introducir ni ejecutar código arbitrario en el entorno de la página.

### Recursos Humanos

Gestión de empleados

[Ver empleados](#) [+ Añadir empleado](#)

#### Añadir empleado

Pavito

Pavito

pavito@paveado.com

Evidencia  | Mitigación   |
|----|--|---------------|---------|---|--|
| 1  | Inyección SQL<br>En endpoint<br>/api/chat                  | Técnica       | Crítico | El servidor ejecuta consultas SQL enviadas por el usuario, como demuestra la petición segura "SELECT "injection_test" AS test, que devuelve el valor sin aplicar ningún filtrado. | Implementar consultas parametrizadas, validar inputs, restringir permisos del usuario SQL y deshabilitar operaciones peligrosas como INTO OUTFILE. |
| 2  | Ausencia de autenticación en la API                        | Lógica        | Alto    | El endpoint permite ejecutar consultas sin token ni credenciales.   | Implementar autenticación obligatoria, tokens JWT con expiración y control de roles.   |
| 3  | Falta de control de autorización (Broken Access Control)   | Lógica        | Alto    | Usuarios sin privilegios pueden solicitar información sensible mediante prompts o consultas SQL indirectas.   | Aplicar RBAC, limitar el acceso por rol y validar permisos antes de ejecutar cualquier consulta.   |
| 4  | Exposición de metadatos del servidor (nginx/1.18.0 Ubuntu) | Configuración | Medio   | La cabecera Server: nginx/1.18.0 (Ubuntu) se expone en respuestas HTTP.   | Ocultar versión del servidor, usar server_tokens off y aplicar hardening en nginx.   |
| 5  | Política CORS permisiva con credenciales                   | Configuración | Alto    | Cabeceras devueltas: Access-Control-Allow-Origin: http://localhost:5173 y Access-Control-Allow-Credentials: true.   | Restringir orígenes válidos, evitar credenciales en CORS y revisar configuración antes de producción   |

6	Falta de protección CSRF	Lógica	Alto	El sistema acepta peticiones automáticas sin token anti-CSRF ni validación de Origin/Referer.	Implementar tokens anti-CSRF, validar cabeceras de origen y de usar cookies con SameSite=Lax o Strict.
7	Falta de rate limiting en la API	Disponibilidad	Medio	Múltiples peticiones consecutivas no generan bloqueo ni limitación.	Implementar rate limiting en proxy inverso o middleware (Nginx, Express-rate-limit).
8	Manejo inseguro de errores	Técnica	Medio	Peticiones mal formadas devuelven mensajes que revelan detalles internos del backend.	Implementar mensajes de error genéricos y registrar detalles solo en logs internos.
9	Ausencia estricta de validación de inputs	Técnica	Alto	El backend acepta cualquier cadena en message, permitiendo manipulación del flujo lógico y del LLM.	Validar inputs con JSON Schema/Zod y limitar el tipo de consultas permitidas.
10	Prompt Injection en el chatbot / IA	Lógica	Alto	El modelo puede ser inducido a ignorar instrucciones y generar consultas SQL más amplias de lo permitido.	Implementar filtros de prompts, sanitización, reglas de seguridad en el generador de SQL y validación posterior del backend.

## 6. Recomendaciones

Las recomendaciones de seguridad se centran en reforzar la protección de la aplicación desde la capa de consultas **SQL** hasta la configuración del servidor. Es fundamental adoptar **parametrización** y **prepared statements**, además de integrar un **ORM** seguro que evite la concatenación de cadenas, con el fin de eliminar el riesgo de inyección. A nivel de servidor, se requiere un proceso de hardening que oculte las versiones de software en las respuestas, configure cabeceras seguras como **X-Content-Type-Options**, **Strict-Transport-Security** y **X-Frame-Options**, y revise la política CORS para restringir los orígenes confiables y evitar exposiciones innecesarias.

En la base de datos, la gestión de roles y permisos debe seguir el principio de mínimo privilegio, segmentando accesos según perfiles de usuario, administradores y servicios. Asimismo, **los datos sensibles deben cifrarse y las contraseñas** almacenarse con algoritmos robustos como **bcrypt** o **argon2**, garantizando que incluso en caso de filtración la información permanezca protegida. Estas medidas reducen la superficie de ataque y aseguran que cada componente tenga únicamente los permisos estrictamente necesarios para su función.

Respecto al chatbot y el modelo de lenguaje, **se recomienda implementar validación y sanitización de prompts**, añadir un middleware de seguridad que filtre las consultas antes de llegar al motor SQL y monitorizar las interacciones para detectar patrones anómalos. Esto permitirá **mitigar riesgos de prompt injection y abusos del LLM**, evitando que un atacante manipule el flujo conversacional para extraer información sensible o ejecutar operaciones indebidas.

Finalmente, para defender la disponibilidad del sistema frente a ataques de denegación de servicio, es necesario configurar límites de conexión y timeouts más estrictos, aplicar rate limiting y reglas específicas en el **WAF**, y considerar mitigación distribuida mediante CDN o balanceadores. En paralelo, se debe **gestionar adecuadamente los endpoints y la exposición pública**, eliminando aquellos de desarrollo, restringiendo el acceso a APIs internas y aplicando autenticación fuerte con **OAuth2** o **JWT** de corta expiración, además de monitorizar accesos y registrar intentos fallidos para detectar actividad sospechosa.

## 7. Conclusiones

El **pentest** realizado demuestra que la aplicación presenta vulnerabilidades críticas en la capa de generación de consultas y en la interacción con el chatbot, lo que permite la ejecución de **SQL** arbitrario y el acceso a datos sensibles. Aunque se han identificado mecanismos defensivos frente a ataques de fuerza bruta y denegación de servicio, la exposición de metadatos, la política CORS insegura y la falta de segmentación de permisos incrementan el riesgo global del sistema.

La principal amenaza radica en la combinación de un modelo de lenguaje con acceso directo a la base de datos, sin controles suficientes de validación y autorización. Esto convierte al chatbot en un vector de ataque capaz de extraer información sensible y manipular la lógica interna.

Se recomienda priorizar la mitigación de las vulnerabilidades críticas (**inyección SQL, acceso indebido a tablas sensibles y prompt injection**), reforzar la configuración del servidor y aplicar controles de seguridad adicionales en la capa de IA. Con estas medidas, la aplicación podrá alcanzar un nivel de seguridad adecuado para su despliegue en un entorno corporativo, garantizando la confidencialidad, integridad y disponibilidad de la información.