AA3 – OWASP Mitigación – Equipo 7

Proyecto: Peluquería Canina API
Integrantes: Estefanía Cale, David Guamán, Stiven Jiménez
Framework: Django REST Framework
Base de datos: SQLite
Autenticación: JWT
Lenguaje: Python
Repositorio: https://github.com/Yamfeild/PelusaSPA.git

1. Resumen técnico del proyecto
La Peluquería Canina API es un sistema backend desarrollado en Django REST Framework que permite la gestión de clientes, peluqueros, mascotas, citas y tarifas mediante una API RESTful.
La aplicación implementa autenticación JWT para garantizar el acceso seguro a los recursos, y organiza sus endpoints bajo la convención /api/<recurso>/.

2.Identificación de al menos 5 vulnerabilidades del OWASP Top 10 (2021) relevantes para su backend.

- A01:2021 - Pérdida de Control de Acceso (Broken Access Control) [1].
- A02:2021 - Fallas Criptográficas (Cryptographic Failures) [1].
- A03:2021 - Inyección (Injection) [1].
- A05:2021 - Configuración de Seguridad Incorrecta (Security Misconfiguration) [1].
- A07:2021 - Fallas de Identificación y Autenticación (Identification and Authentication Failures) [1].

3.Descripción del riesgo y del posible impacto de cada vulnerabilidad en su sistema.

OWASP(2021)	Descripción [5]	Ejemplo de riesgo en el proyecto	Impacto
A01:2021: - Pérdida de Control de Acceso	Ocurre cuando el sistema no restringe adecuadamente lo que los usuarios autenticados pueden hacer.[5] Esto incluye acceso a recursos ajenos, manipulación de IDs en las rutas o	Un cliente autenticado podría intentar acceder a /api/citas/5/ aunque esa cita pertenezca a otro usuario, si el backend no valida la propiedad del recurso.	Exposición o modificación de información privada (citas, mascotas o datos de otros clientes).

	ejecución de acciones sin autorización.		
A02:2021:- Fallas Criptográficas	Ocurre cuando datos externos no validados se envían directamente a una consulta SQL, comando del sistema o intérprete.	Si se construyeran consultas SQL manuales con concatenación de texto: cursor.execute("SELECT * FROM duenos WHERE nombre = '" + nombre + "'")	Modificación, eliminación o robo de datos desde la base de datos.
A03:2021: - Inyección	Incluye configuraciones inseguras del servidor, permisos excesivos, cabeceras HTTP ausentes o parámetros sensibles expuestos.	Tener DEBUG=True en producción, CORS abierto a todos los orígenes o exponer la SECRET_KEY en el código.	Permite obtener información sensible del sistema o ejecutar ataques CSRF.
A05:2021:- Configuración de Seguridad Incorrecta	Se produce cuando los mecanismos de autenticación son débiles o mal implementados: sesiones sin expiración, contraseñas poco seguras, validaciones ausentes, etc.	JWT sin expiración o sin verificación de firma podría permitir que usuarios accedan indefinidamente	Suplantación de identidad o sesiones activas de forma indefinida.
A07:2021: - Fallas de Identificación y Autenticación	Surge cuando los datos sensibles (contraseñas, tokens, información personal) no se protegen adecuadamente mediante cifrado o hashing.	Guardar contraseñas sin cifrar en la base de datos o transmitir credenciales por HTTP no seguro.	Compromiso de cuentas, robo de datos o acceso no autorizado.

4.Medidas de mitigación implementadas o planificadas, explicando cómo se aplicaron en el código o la configuración.

Para mitigar estos riesgos, es fundamental integrar la seguridad.

- Implementación de CORS (Cross-Origin Resource Sharing): Se configuró el middleware CORS para controlar qué dominios pueden realizar peticiones al

servidor. Esto evita ataques del tipo Cross-Site Request Forgery (CSRF) y accesos no autorizados desde orígenes externos [4].

- Autenticación mediante tokens JWT: protege las rutas y verificar la identidad del usuario en cada solicitud. Cada vez que un usuario inicia sesión correctamente, el sistema genera un token firmado que incluye datos como el identificador del usuario y su rol. En las rutas protegidas, el servidor valida el token antes de permitir el acceso, garantizando que solo usuarios autenticados puedan consumir los recursos [3].

- Uso de códigos de estado HTTP para manejo seguro de errores: Se estableció una política de respuestas controladas según el tipo de error, sin revelar detalles del sistema ni de la base de datos. Esto evita que atacantes obtengan información interna o estructura del backend [2].

5.Evidencias de verificación:capturas de pruebas, fragmentos de código o resultados de herramientas de análisis.

Configuración de Cors:

```
INSTALLED_APPS = [
    'corsheaders', #clave para CORS
]

MIDDLEWARE = [
    'corsheaders.middleware.CorsMiddleware', #clave para CORS
]

CORS_ALLOWED_ORIGINS = [
    "http://localhost:3000",
]

CORS_ALLOW_ALL_ORIGINS = True  #clave para CORS
```

Configuración de JWT:

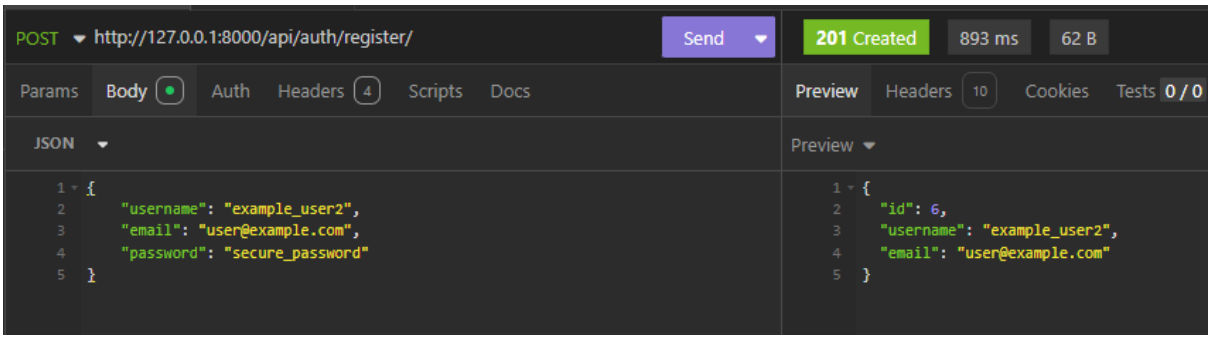
```
INSTALLED_APPS = [
    'rest_framework_simplejwt'
]
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'rest_framework_simplejwt.authentication.JWTAuthentication',
    )
}
```

Uso de errores http:

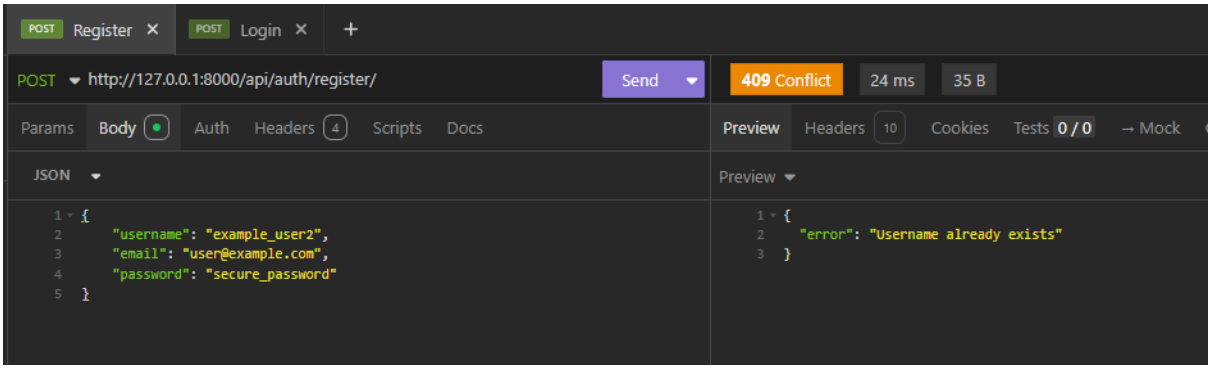
```python
        username = request.data.get('username')
        email = request.data.get('email')

        # Verificar si el usuario ya existe
        if User.objects.filter(username=username).exists():
            return Response(
                {'error': 'Username already exists'},
                status=status.HTTP_409_CONFLICT
            )

        # Verificar si el email ya existe
        if User.objects.filter(email=email).exists():
            return Response(
                {'error': 'Email already registered'},
                status=status.HTTP_409_CONFLICT
            )
```

Análisis de prueba con insomnia:

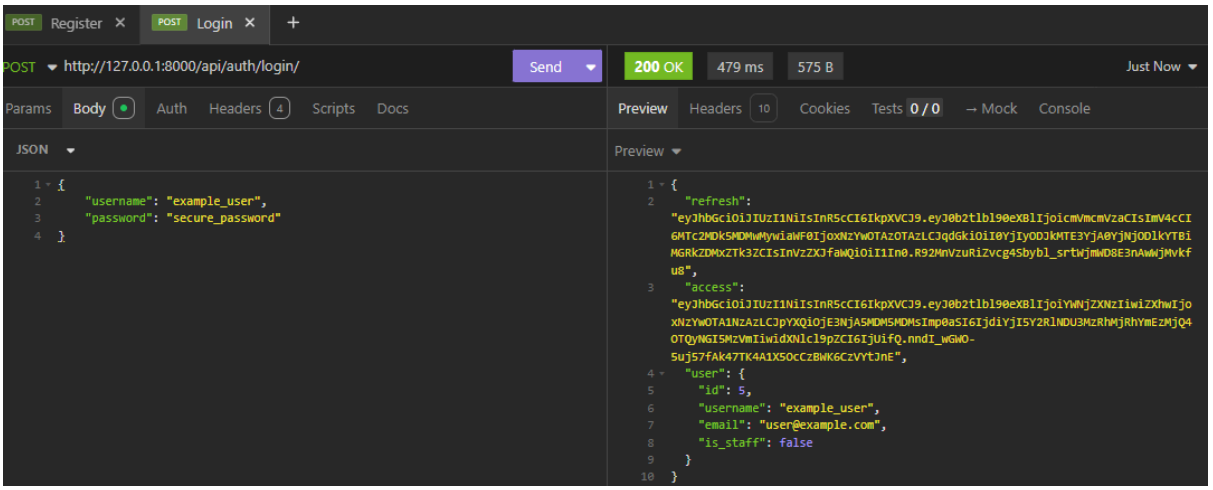
Registro de un nuevo usuario:



Registro de Usuario si tiene el mismo correo:



Inicio de sesión exitoso:



6.Reflexión personal:qué aprendió sobre seguridad durante esta unidad y cómo mejoraría el diseño futuro del sistema.

Durante el desarrollo del backend nos dimos cuenta y aprendimos que esta no es una etapa final sino un proceso continuo que ayuda a mejorar la seguridad de nuestros programas, al aplicar medias **OWASP 2021** comprendemos la gran importancia de proteger el sistema, permitiendo mejorar la seguridad y la confianza de los datos.

Preguntas orientadoras

1.¿Qué vulnerabilidad del OWASP Top 10 representa el mayor riesgo para tu proyecto y por qué?

La vulnerabilidad que representa el mayor riesgo para el proyecto es A07:2021 - Fallas de Identificación y Autenticación (Identification and Authentication Failures). Esto se debe a que el sistema maneja usuarios, inicios de sesión y autenticación mediante tokens, por lo que una configuración inadecuada o una verificación incorrecta del token podría permitir el acceso no autorizado a información sensible o funcionalidades restringidas.

2.¿Cómo validan que las medidas implementadas realmente mitigan la vulnerabilidad detectada?

La validación se realizó utilizando Insomnia para probar los endpoints del sistema. Se enviaron solicitudes desde distintos orígenes y con diferentes condiciones para verificar el funcionamiento de las medidas implementadas:

- CORS: se comprobó que solo los dominios permitidos pueden acceder; otros fueron bloqueados con respuesta 403 Forbidden.
- JWT: se probaron tokens válidos, inválidos y ausentes, confirmando que el sistema devuelve 401 Unauthorized cuando el token no es válido y 200 OK solo con autenticación correcta.

- Códigos HTTP: se verificó que ante errores de solicitud se devuelvan respuestas controladas (400, 403, 404, 500) sin revelar información sensible.

Estas pruebas demostraron que las configuraciones de CORS, JWT y el manejo de códigos de error REST mitigan efectivamente las vulnerabilidades detectadas en el sistema.

3.¿Qué relación encuentras entre las vulnerabilidades OWASP y los componentes del modelo C4 (backend, base de datos, API Gateway)?
Las vulnerabilidades OWASP no afectan solo una parte, sino que se distribuyen entre los niveles que se encuentran en el modelo C4:

- El **backend** debe implementar los controles de acceso, validaciones y autenticación.
- La **base de datos** requiere cifrado y consultas parametrizadas.
- El **gateway** o capa de comunicación debe aplicar configuraciones seguras y validar los tokens entrantes.

En conjunto, estas capas garantizan la confidencialidad, integridad y disponibilidad del sistema.

BIBLIOGRAFÍA

[1]“Home - OWASP Top 10:2021”, Owasp.org. [En línea]. Disponible en: https://owasp.org/Top10/es.
[2]“Códigos de estado de respuesta HTTP”, MDN Web Docs. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Web/HTTP/Reference/Status.
[3]“CICS Transaction Server for z/OS”, Ibm.com, 05-sep-2025. [En línea]. Disponible en: https://www.ibm.com/docs/es/cics-ts/6.x?topic=cics-json-web-token-jwt.
[4]“Intercambio de recursos de origen cruzado (CORS)”, MDN Web Docs. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Web/HTTP/Guides/CORS.
[5] H. R. G. Brito, R. A. A. Martínez, and D. G. Reyes, "Evaluación de los mecanismos de autenticación en aplicaciones web con la Guía de Pruebas de Seguridad de OWASP," Evaluation of the authentication mechanisms in web applications with the OWASP Testing Guide.