Informe de Análisis de Seguridad - Aplicación Médica

Fecha de análisis: 2025-11-29

1. Descripción Breve de la Aplicación Analizada

1.1. Propósito y Funcionalidad

La aplicación es una herramienta web monousuario diseñada para el registro personal de peso, talla y cálculo del Índice de Masa Corporal (IMC). Su objetivo principal es permitir a un único usuario realizar un seguimiento de su peso corporal y obtener información sobre su estado nutricional mediante el cálculo automático del IMC.

1.2. Características Principales

• Registro de datos personales: Nombre, apellidos, fecha de nacimiento y talla (en metros)
• Registro de peso: Permite registrar el peso actual con fecha y hora automáticas
• Cálculo automático de IMC: Calcula y muestra el Índice de Masa Corporal basado en el último peso registrado
• Estadísticas históricas: Muestra número de pesajes, peso máximo y peso mínimo registrados
• Sincronización bidireccional: Entre frontend (localStorage) y backend (memoria)
• Modo offline: Funciona sin conexión al servidor utilizando almacenamiento local

1.3. Arquitectura Técnica

• Backend: Flask (Python) con API REST
• Frontend: JavaScript vanilla con almacenamiento en localStorage
• Almacenamiento: Memoria en backend + localStorage en frontend
• Tests: 86 tests backend (pytest) + ~66 tests frontend (Jest)
• Gestión de vulnerabilidades: DefectDojo integrado para seguimiento de debilidades de seguridad

1.4. Análisis de Datos que Maneja la Aplicación

La aplicación maneja diferentes tipos de datos que requieren diferentes niveles de protección:

1.4.1. Datos Sensibles (Personales de Salud)

• Peso corporal (kg): Dato biométrico personal
• Talla/Altura (m): Dato biométrico personal
• Fecha de nacimiento: Permite inferir edad y otros datos demográficos
• Nombre completo: Identificador personal
• Apellidos: Identificador personal
Clasificación según RGPD: Estos datos están categorizados como datos personales sensibles según el Reglamento General de Protección de Datos, ya que los datos de salud (peso, altura, IMC) están incluidos en la categoría de datos especiales.

Almacenamiento actual:

- Frontend: localStorage del navegador (cliente)

- Backend: Memoria (volátil, se pierde al reiniciar)

1.4.2. Datos Internos

• ID de usuario: Constante (USER_ID = 1) - Monousuario
• Fecha y hora de registros de peso: Metadatos de los pesajes

• Cálculos derivados: IMC, estadísticas (peso máximo, mínimo, número de pesajes)

1.4.3. Datos Públicos

• Interfaz de usuario: Textos de la aplicación, mensajes, traducciones
• Configuración de límites de validación: Límites de peso y altura permitidos

1.5. Contexto de Seguridad

La aplicación maneja datos de salud personales (peso, altura, fecha de nacimiento), aunque en un contexto monousuario y sin autenticación compleja. No almacena datos médicos críticos ni información de identificación sensible como DNI o números de seguridad social, pero sí información personal que requiere protección adecuada según estándares de protección de datos.

2. Identificación de Vectores de Ataque Potenciales

2.0. Funcionalidades Susceptibles a Ataques

La aplicación incluye las siguientes funcionalidades que pueden ser susceptibles a ataques:

2.0.1. Formularios de Registro de Datos

• Formulario de perfil de usuario: Captura nombre, apellidos, fecha de nacimiento y talla
• Formulario de registro de peso: Captura peso actual con validación de variación diaria
• Vector de ataque: Inyección de datos maliciosos, validación de tipos insuficiente

2.0.2. API REST sin Autenticación

• Endpoints públicos: /api/user, /api/weight, /api/imc, /api/stats
• Métodos HTTP: GET, POST sin autenticación
• Vector de ataque: Acceso no autorizado, manipulación de datos, CSRF

2.0.3. Almacenamiento Local (localStorage)

• Datos en cliente: Todos los datos se almacenan en localStorage del navegador
• Vector de ataque: Acceso a datos si el dispositivo es comprometido, XSS que puede leer/escribir localStorage

2.0.4. Cálculos Matemáticos (IMC)

• Fórmula IMC: peso / (altura²)
• Vector de ataque: Valores NaN, Infinity que pueden causar errores o resultados incorrectos

2.0.5. Integración con DefectDojo

• Endpoints de importación/exportación: /api/defectdojo/export-dump, /api/defectdojo/import-dump
• Vector de ataque: Ejecución de comandos del sistema, manipulación de base de datos
Basado en el análisis CWE-699 (Software Development) realizado, se han identificado los siguientes vectores de ataque potenciales:

2.1. Vectores de Ataque Identificados

2.1.1. Inyección de Datos Maliciosos (Type Confusion)

CWE-1287: Improper Validation of Specified Type of Input

Ubicación:

- Backend: app/routes.py (conversiones de float())

- Frontend: app/static/js/main.js, app/static/js/storage.js (conversiones de parseFloat())

Vector de ataque:

- Envío de valores no numéricos que se convierten a NaN o Infinity
- Propagación de valores inválidos en cálculos matemáticos (IMC)
- Causa de errores en la aplicación o resultados incorrectos

Ejemplo de ataque:

// Resultado: parseFloat("NaN") = NaN

// IMC calculado con NaN = NaN (resultado inválido)

2.1.2. Clickjacking (UI Redressing)

CWE-1021: Improper Restriction of Rendered UI Layers or Frames

Estado: ■ RESUELTO - Headers de seguridad implementados

Ubicación: app/__init__.py (líneas 28-36)

Vector de ataque original:

- Inclusión de la aplicación en un iframe malicioso
- Superposición de elementos invisibles sobre la interfaz
- Manipulación de acciones del usuario sin su conocimiento

Mitigación implementada:

- ■ X-Frame-Options: DENY - Previene que la aplicación se muestre en iframes
- ■ Content-Security-Policy: frame-ancestors 'none' - Política adicional contra frames
- ■ X-Content-Type-Options: nosniff - Previene MIME type sniffing
- ■ X-XSS-Protection: 1; mode=block - Protección adicional contra XSS

Ejemplo de ataque previo:

Click aquí

(Ahora bloqueado por los headers de seguridad)

2.1.3. Cross-Origin Resource Sharing (CORS) Permisivo

CWE-942: Overly Permissive Cross-domain Whitelist

Ubicación: app/__init__.py (configuración CORS con origins: "*")

Vector de ataque:

- Cualquier sitio web puede realizar peticiones a la API
- Aumento de la superficie de ataque si el backend se expone fuera de localhost
- Posibilidad de realizar ataques CSRF desde dominios externos

Ejemplo de ataque:

```
fetch('http://aplicacion-medica.com/api/user', {
method: 'POST',
headers: {'Content-Type': 'application/json'},
body: JSON.stringify({nombre: 'Ataque', talla_m: 'NaN'})
});
```

2.1.4. Manejo Inadecuado de Excepciones

CWE-703: Improper Check or Handling of Exceptional Conditions

Ubicación: app/routes.py (uso de Exception genérico)

Vector de ataque:

- Ocultación de errores inesperados que podrían indicar vulnerabilidades
- Dificultad para detectar y responder a ataques
- Información de error insuficiente para debugging y monitoreo

2.1.5. Validación de Entrada Insuficiente (Resuelto)

CWE-20: Improper Input Validation

Estado: ■ RESUELTO - Se implementó validación robusta de nombres con sanitización

Ubicación: app/helpers.py, app/routes.py, app/static/js/main.js

Vector de ataque original:

- Inyección de caracteres peligrosos (< > " ')
- Ataques XSS potenciales
- Corrupción de datos almacenados

3. Análisis de Riesgos Asociados

3.1. Matriz de Riesgos

CWE	Vector de Ataque	Probabilidad	Impacto	Riesgo	Estado
CWE-1287	Type Confusion (Backend)	Media	Medio	MEDIO	■■ Pendiente
CWE-843	Type Confusion (Frontend)	Media	Medio	MEDIO	■■ Pendiente
CWE-1021	Clickjacking	Baja	Medio	MEDIO	■ Resuelto
CWE-942	CORS Permisivo	Baja*	Alto*	MEDIO/ALTO*	■■ Pendiente
CWE-703	Manejo de Excepciones	Baja	Bajo	BAJO	■■ Mejorado parcialmente
CWE-20	Validación de Entrada	Media	Medio	MEDIO	■ Resuelto

*Riesgo aumenta significativamente si el backend se expone fuera de localhost

3.2. Análisis Detallado de Riesgos

3.2.1. Riesgo MEDIO: Type Confusion (CWE-1287, CWE-843)

Descripción del riesgo:

- Los valores NaN e Infinity pueden propagarse en cálculos matemáticos
- Resultados de IMC incorrectos o inválidos
- Posible corrupción de datos almacenados

Impacto en el negocio:

- Integridad de datos: Los datos almacenados pueden volverse inválidos
- Confiabilidad: Los usuarios pueden recibir información incorrecta sobre su IMC

- Experiencia de usuario: Errores en la aplicación que afectan la usabilidad

Mitigación actual:

- ■ Validaciones defensivas antes de calcular IMC (backend y frontend)
- ■■ Falta validación de NaN e Infinity después de conversiones

Recomendación:

- Implementar validación de tipos antes de conversión
- Verificar que los números sean finitos después de float() y parseFloat()

3.2.2. Riesgo MEDIO: Clickjacking (CWE-1021) - RESUELTO

Descripción del riesgo (anterior):

- La aplicación puede ser embebida en iframes maliciosos
- Los usuarios pueden ser engañados para realizar acciones no deseadas
- Posible robo de datos personales mediante superposición de elementos

Impacto en el negocio (anterior):

- Confidencialidad: Los datos personales pueden ser comprometidos
- Integridad: Los datos pueden ser modificados sin consentimiento del usuario
- Reputación: Pérdida de confianza si se explota esta vulnerabilidad

Mitigación implementada:

- ■ X-Frame-Options: DENY - Implementado en app/__init__.py
- ■ Content-Security-Policy: frame-ancestors 'none' - Implementado
- ■ X-Content-Type-Options: nosniff - Implementado
- ■ X-XSS-Protection: 1; mode=block - Protección adicional

Estado: ■ VULNERABILIDAD RESUELTA - Los headers de seguridad protegen contra clickjacking

3.2.3. Riesgo MEDIO/ALTO: CORS Permisivo (CWE-942)

Descripción del riesgo:

- Cualquier sitio web puede realizar peticiones a la API
- Aumento significativo de la superficie de ataque en producción
- Posibilidad de ataques CSRF desde dominios externos

Impacto en el negocio:

- Confidencialidad: Datos pueden ser accedidos desde sitios maliciosos
- Integridad: Datos pueden ser modificados mediante ataques CSRF
- Disponibilidad: Posible denegación de servicio mediante peticiones masivas

Mitigación actual:

- ■■ CORS configurado con origins: "*" (permite cualquier origen)
- ■ Actualmente solo accesible desde localhost (riesgo limitado)

Recomendación:

- Restringir CORS a dominios específicos cuando se exponga el backend
- Implementar validación de origen en producción

- Considerar implementar tokens CSRF para operaciones sensibles

3.2.4. Riesgo BAJO: Manejo de Excepciones (CWE-703)

Descripción del riesgo:

- Uso de Exception genérico puede ocultar errores inesperados
- Dificultad para detectar y responder a vulnerabilidades
- Información de error insuficiente para monitoreo

Impacto en el negocio:

- Mantenibilidad: Dificultad para identificar y corregir problemas
- Seguridad: Errores de seguridad pueden pasar desapercibidos

Mitigación actual:

- ■ Validación de nombres usa manejo estructurado de errores
- ■■ Conversiones de float() aún usan Exception genérico

Recomendación:

- Especificar excepciones concretas (ValueError, TypeError, KeyError)
- Implementar logging detallado de errores

3.3. Resumen de Riesgos por Categoría

Riesgos Críticos: Ninguno identificado

Riesgos Altos:

- CWE-942 (CORS) - Solo si se expone fuera de localhost

Riesgos Medios:

- CWE-1287 (Type Confusion Backend)
- CWE-843 (Type Confusion Frontend)

Riesgos Bajos:

- CWE-703 (Manejo de Excepciones)

Riesgos Resueltos:

- CWE-20 (Validación de Entrada) ■

4. Nivel ASVS Seleccionado, con Justificación

4.1. OWASP Application Security Verification Standard (ASVS)

El OWASP Application Security Verification Standard (ASVS) es un estándar de seguridad para aplicaciones web que define tres niveles de verificación:

• Nivel 1 (Básico): Para aplicaciones de bajo riesgo, con controles de seguridad básicos
• Nivel 2 (Estándar): Para aplicaciones de riesgo estándar, con controles de seguridad estándar
• Nivel 3 (Avanzado): Para aplicaciones de alto riesgo, con controles de seguridad avanzados

4.2. Nivel Seleccionado: NIVEL 2 (ESTÁNDAR)

4.3. Justificación de la Selección

4.3.1. Naturaleza de los Datos

La aplicación maneja datos de salud personales (peso, altura, fecha de nacimiento), aunque no sean datos médicos críticos ni información de identificación sensible. Según el Reglamento General de Protección de Datos (RGPD), los datos de salud están categorizados como datos personales sensibles, lo que requiere un nivel de protección superior al básico.

Justificación: Aunque los datos no son críticos, el hecho de ser datos de salud justifica un nivel de seguridad estándar (Nivel 2) en lugar de básico (Nivel 1).

4.3.2. Contexto de Uso

• Monousuario: La aplicación es monousuario, lo que reduce el riesgo de exposición de datos entre usuarios
• Sin autenticación compleja: No requiere autenticación robusta, lo que limita algunos vectores de ataque
• Almacenamiento local: Los datos se almacenan principalmente en localStorage, reduciendo el riesgo de exposición en el servidor
Justificación: El contexto monousuario y el almacenamiento local reducen algunos riesgos, pero no eliminan la necesidad de controles de seguridad estándar.

4.3.3. Vectores de Ataque Identificados

El análisis CWE-699 identificó 6 debilidades de seguridad, de las cuales:

- 1 está resuelta (CWE-20)

- 4 requieren atención (CWE-1287, CWE-843, CWE-1021, CWE-942)

- 1 está mejorada parcialmente (CWE-703)

Justificación: La presencia de múltiples debilidades de seguridad de severidad media justifica la implementación de controles de seguridad estándar (Nivel 2) en lugar de básicos (Nivel 1).

4.3.4. Requisitos de Cumplimiento

Aunque la aplicación no maneja datos médicos críticos, el manejo de datos de salud personales puede requerir cumplimiento con:

- RGPD: Reglamento General de Protección de Datos

- Estándares de seguridad de aplicaciones web: Buenas prácticas de la industria

Justificación: El cumplimiento con estándares de seguridad de la industria y protección de datos personales requiere al menos un nivel estándar de seguridad (Nivel 2).

4.3.5. Por qué NO Nivel 3 (Avanzado)

• No maneja datos médicos críticos
• No requiere autenticación multi-factor
• No tiene requisitos de alta disponibilidad
• No maneja transacciones financieras
• No es una aplicación crítica para la vida
Justificación: La aplicación no cumple con los criterios para requerir controles de seguridad avanzados (Nivel 3).

4.3.6. Por qué NO Nivel 1 (Básico)

• Maneja datos de salud personales (datos sensibles según RGPD)
• Identificadas múltiples debilidades de seguridad de severidad media
• Requiere protección contra vectores de ataque comunes (clickjacking, CORS, type confusion)
Justificación: La naturaleza de los datos y los vectores de ataque identificados requieren controles de seguridad más allá del nivel básico.

4.4. Enumeración de Requerimientos ASVS Nivel 2

Basado en el análisis realizado, se enumeran los requerimientos ASVS Nivel 2 relevantes para esta aplicación. El estándar ASVS define 14 categorías de verificación (V1-V14). Se detallan las aplicables:

4.4.1. V1: Architecture, Design and Threat Modeling

Requerimientos ASVS Nivel 2 aplicables:

- V1.1: ■ Arquitectura de seguridad definida y documentada

- V1.2: ■ Análisis de amenazas realizado (análisis CWE-699 completado)

- V1.3: ■ Principios de seguridad aplicados (defensa en profundidad)

- V1.4: ■ Separación de responsabilidades (frontend/backend/almacenamiento)

Estado de cumplimiento: ■ CUMPLE - Arquitectura clara y análisis de amenazas documentado

4.4.2. V2: Authentication

Requerimientos ASVS Nivel 2 aplicables:

- ■■ No aplicable - Aplicación monousuario sin autenticación compleja

Justificación: La aplicación está diseñada como herramienta personal monousuario, sin sistema de autenticación. El usuario único se identifica mediante USER_ID constante.

4.4.3. V3: Session Management

Requerimientos ASVS Nivel 2 aplicables:

- ■■ No aplicable - La aplicación no utiliza sesiones en el servidor

Justificación: Al ser monousuario y almacenar datos principalmente en localStorage del cliente, no se requieren sesiones del lado del servidor.

4.4.4. V4: Access Control

Requerimientos ASVS Nivel 2 aplicables:

- ■■ No aplicable - Aplicación monousuario sin control de acceso entre usuarios

Justificación: Al existir un único usuario (USER_ID = 1), no hay necesidad de control de acceso basado en roles o usuarios.

4.4.5. V5: Validation, Sanitization and Encoding

Requerimientos ASVS Nivel 2 aplicables:

- V5.1: ■ Validación de entrada en todas las fuentes de datos (frontend y backend)

- V5.2: ■ Validación de tipos de datos (nombres, números, fechas)

- V5.3: ■ Sanitización de entrada (función validate_and_sanitize_name() implementada)

- V5.4: ■■ Validación de tipos numéricos mejorable (CWE-1287, CWE-843 pendientes)

- V5.5: ■ Validación de límites de rango (altura: 0.4-2.72m, peso: 2-650kg)

- V5.6: ■ Validación de formato (fechas en formato ISO)

Estado de cumplimiento: ■■ PARCIAL - Validación robusta implementada, pero falta validación de NaN/Infinity

4.4.6. V6: Stored Cryptographically Sensitive Data

Requerimientos ASVS Nivel 2 aplicables:

- ■■ No aplicable - Datos almacenados localmente en cliente, no en servidor

Justificación: Los datos sensibles se almacenan en localStorage del navegador (cliente). El backend solo mantiene datos en memoria volátil.

4.4.7. V7: Error Handling and Logging

Requerimientos ASVS Nivel 2 aplicables:

- V7.1: ■■ Manejo estructurado de errores (mejorado en validación de nombres, pendiente en conversiones)

- V7.2: ■ Códigos de error HTTP apropiados (400, 404, 500)

- V7.3: ■■ Logging de errores mejorable (CWE-703 mejorado parcialmente)

- V7.4: ■ No se exponen detalles técnicos de errores al usuario final

Estado de cumplimiento: ■■ PARCIAL - Mejorado en validación de nombres, falta especificar excepciones en conversiones

4.4.8. V8: Data Protection

Requerimientos ASVS Nivel 2 aplicables:

- V8.1: ■ Validaciones defensivas implementadas (antes de cálculos críticos)

- V8.2: ■ Headers de seguridad implementados (X-Frame-Options, CSP, etc.)

- V8.3: ■ Protección contra clickjacking (CWE-1021 resuelto)

- V8.4: ■■ CORS demasiado permisivo (CWE-942 pendiente para producción)

Estado de cumplimiento: ■ BUENO - Headers de seguridad implementados, CORS pendiente de restricción en producción

4.4.9. V9: Communications

Requerimientos ASVS Nivel 2 aplicables:

- V9.1: ■ HTTPS recomendado para producción (actualmente HTTP en desarrollo)

- V9.2: ■■ CORS configurado pero permisivo (origins: "*")

- V9.3: ■■ No se implementa validación de origen específico

Estado de cumplimiento: ■■ PARCIAL - CORS funciona pero es demasiado permisivo para producción

4.4.10. V10: Malicious Code

Requerimientos ASVS Nivel 2 aplicables:

- V10.1: ■ Validación y sanitización de entrada implementada

- V10.2: ■ Protección contra XSS mediante sanitización de nombres

- V10.3: ■ Headers X-XSS-Protection implementados

- V10.4: ■ No se ejecuta código no confiable

Estado de cumplimiento: ■ CUMPLE - Validación y sanitización robustas implementadas

4.4.11. V11: Business Logic

Requerimientos ASVS Nivel 2 aplicables:

- V11.1: ■ Validación de reglas de negocio (variación de peso máximo por día)

- V11.2: ■ Validación de límites de negocio (peso, altura en rangos razonables)

Estado de cumplimiento: ■ CUMPLE - Reglas de negocio validadas

4.4.12. V12: Files and Resources

Requerimientos ASVS Nivel 2 aplicables:

- ■■ Parcialmente aplicable - Endpoints de importación/exportación de DefectDojo

- V12.1: ■ Validación de tipo de archivo (solo .sql permitido)

- V12.2: ■ Uso de secure_filename() para nombres de archivo

Estado de cumplimiento: ■ CUMPLE - Validaciones de archivo implementadas

4.4.13. V13: API

Requerimientos ASVS Nivel 2 aplicables:

- V13.1: ■ API REST documentada mediante código

- V13.2: ■ Validación de entrada en todos los endpoints

- V13.3: ■■ Sin autenticación/autorización (no aplicable por diseño monousuario)

- V13.4: ■ Manejo de errores con códigos HTTP apropiados

Estado de cumplimiento: ■ BUENO - API bien estructurada con validación, sin autenticación por diseño

4.4.14. V14: Configuration

Requerimientos ASVS Nivel 2 aplicables:

- V14.1: ■ Configuración centralizada (app/config.py)

- V14.2: ■ Headers de seguridad configurados

- V14.3: ■■ CORS configurado pero permisivo en desarrollo

Estado de cumplimiento: ■ CUMPLE - Configuración centralizada y bien estructurada

4.5. Conclusión del Nivel ASVS

Nivel seleccionado: NIVEL 2 (ESTÁNDAR)

Justificación resumida:

1. Maneja datos de salud personales (datos sensibles según RGPD)

2. Identificadas múltiples debilidades de seguridad de severidad media

3. Requiere controles de seguridad estándar para proteger datos personales

4. No requiere controles avanzados (Nivel 3) por no manejar datos críticos

Estado de cumplimiento: ■ BUENO - EN PROGRESO - Se han implementado mejoras significativas (headers de seguridad, validación robusta). Faltan mejoras menores en validación de tipos y CORS para alcanzar el cumplimiento completo del Nivel 2.

5. Conclusión Final

5.1. Resumen Ejecutivo

La aplicación médica analizada es una herramienta web monousuario para el registro personal de peso e IMC. Aunque no maneja datos médicos críticos, sí procesa datos de salud personales que requieren protección adecuada según estándares de seguridad y protección de datos.

El análisis de seguridad basado en CWE-699 (Software Development) identificó 6 debilidades de seguridad, de las cuales:

- 2 están completamente resueltas (CWE-20: Validación de entrada, CWE-1021: Clickjacking)

- 3 requieren atención (CWE-1287, CWE-843, CWE-942)

- 1 está mejorada parcialmente (CWE-703: Manejo de excepciones)

5.2. Vectores de Ataque Principales

Los principales vectores de ataque identificados son:

• Type Confusion (CWE-1287, CWE-843): Validación insuficiente de tipos numéricos que puede resultar en valores NaN o Infinity en cálculos
• CORS Permisivo (CWE-942): Configuración que permite peticiones desde cualquier origen, aumentando la superficie de ataque
Vectores de ataque resueltos:

- ■ Clickjacking (CWE-1021): Headers de seguridad implementados (X-Frame-Options, CSP, etc.)

5.3. Nivel de Riesgo General

El riesgo general de la aplicación es MEDIO, con los siguientes factores:

• Riesgos Críticos: Ninguno identificado
• Riesgos Altos: CWE-942 (solo si se expone fuera de localhost)
• Riesgos Medios: CWE-1287, CWE-843
• Riesgos Bajos: CWE-703
• Riesgos Resueltos: CWE-20 (Validación de entrada), CWE-1021 (Clickjacking)

5.4. Nivel ASVS Recomendado

Se recomienda implementar controles de seguridad según OWASP ASVS Nivel 2 (Estándar) debido a:

• Naturaleza de los datos: Datos de salud personales (datos sensibles según RGPD)
• Vectores de ataque identificados: Múltiples debilidades de severidad media
• Requisitos de cumplimiento: Estándares de seguridad de la industria y protección de datos
Estado de cumplimiento: ■■ PARCIAL - Se requiere implementar mejoras para alcanzar el cumplimiento completo.

5.5. Recomendaciones Prioritarias

Para alcanzar el cumplimiento del ASVS Nivel 2, se recomienda implementar las siguientes mejoras en orden de prioridad:

Prioridad Alta:

• Validar NaN e Infinity en frontend y backend (CWE-1287, CWE-843)
• Validar tipos antes de conversión
• Verificar que los números sean finitos después de float() y parseFloat()

Prioridad Media:

• Restringir CORS en producción (CWE-942)

• Limitar orígenes permitidos cuando el backend se exponga fuera de localhost
Implementar validación de origen

Mejorar manejo de excepciones (CWE-703)

• Especificar excepciones concretas en lugar de Exception genérico
• Implementar logging detallado de errores

5.6. Evaluación Final

La aplicación implementa buenas prácticas de seguridad en validación de entrada, headers de seguridad y manejo defensivo de datos. Las mejoras implementadas incluyen:

• ■ Validación robusta de nombres con sanitización
• ■ Headers de seguridad contra clickjacking
• ■ Validaciones defensivas antes de cálculos críticos
• ■ Arquitectura bien estructurada con análisis de amenazas
Sin embargo, aún requiere mejoras en validación de tipos numéricos (NaN/Infinity) y restricción de CORS en producción para alcanzar el cumplimiento completo del ASVS Nivel 2.

Estado general: ■ BUENO - EN PROGRESO HACIA CUMPLIMIENTO COMPLETO

Las debilidades restantes son de severidad media o baja y pueden corregirse sin cambios arquitectónicos significativos, lo que facilita su implementación y mejorará aún más la postura de seguridad de la aplicación. El progreso realizado muestra un compromiso con la seguridad y buenas prácticas de desarrollo.

Fecha de informe: 2025-11-29

- CWE-699: Software Development

- OWASP ASVS

- OWASP Top 10