Pruebas de Software Servicio OpenAI SmishGuard

1. Introducción

Propósito

Este documento describe las pruebas necesarias para validar el servicio Flask que utiliza la API de OpenAI para analizar mensajes de texto, evaluando su seguridad contra posibles amenazas de phishing. El objetivo de estas pruebas es asegurar la funcionalidad, desempeño y estabilidad del servicio bajo distintas condiciones de carga.

Alcance de las pruebas

Las pruebas cubrirán los siguientes tipos de validación:

- Pruebas Unitarias
- Pruebas de Carga
- Pruebas de API

Objetivos

1. Asegurar que los endpoints respondan correctamente bajo distintas entradas y condiciones.
2. Validar el desempeño del servicio bajo condiciones de carga.
3. Identificar y documentar cualquier fallo crítico.

2. Metodología de Pruebas

Tipos de pruebas

- **Pruebas Unitarias**: Valida la funcionalidad de cada endpoint y funciones internas.
- **Pruebas de Carga**: Analiza el rendimiento bajo carga usando `Locust`.
- **Pruebas de API**: Realiza pruebas en los endpoints para verificar el manejo adecuado de entradas y respuestas de OpenAI.

Herramientas de prueba

- **Unittest** y **Pytest** para pruebas unitarias.
- **Locust** para pruebas de carga.
- **Mocking** para simular respuestas de OpenAI en pruebas de API e integración.

Criterios de aceptación

- Cada endpoint debe responder con el código de estado adecuado (200, 400, 500).

- El análisis de mensajes debe cumplir el formato JSON especificado.
- Respuesta de OpenAI simulada y manejada de forma correcta en caso de errores.

3. Pruebas Unitarias

Propósito

Validar la funcionalidad individual de cada endpoint en el servicio, garantizando respuestas adecuadas ante distintos escenarios.

Herramientas

`Unittest`, `Pytest`

Estrategia de Pruebas

Las pruebas cubrirán el flujo de cada endpoint, simulando respuestas y errores de OpenAI para asegurar que la lógica del servicio se mantenga consistente.

Cobertura de Pruebas

Se espera una cobertura del 90% en los endpoints principales y las funciones críticas de lógica.

Criterios de Ejecución

Cada prueba unitaria se ejecutará usando un cliente de pruebas Flask en un entorno controlado.

Casos de Prueba

`GET /`

- **Descripción**: Verifica que el endpoint raíz responda con "Hello, world!".
- **Entradas**: Ninguna.
- **Acciones**: Realizar un `GET /`.
- **Resultados Esperados**: Respuesta con código 200 y mensaje "Hello, world!".

`GET /ping`

- **Descripción**: Valida la conectividad del servicio.
- **Entradas**: Ninguna.
- **Acciones**: Realizar un `GET /ping`.
- **Resultados Esperados**: Respuesta con código 200 y `{"message": "pong"}`.

`POST /consultar-modelo-gpt`

- **Descripción**: Envía un mensaje de prueba para análisis de phishing.
- **Entradas**: JSON con campo `mensaje`.
- **Acciones**: Enviar `POST` con el mensaje.
- **Resultados Esperados**: Respuesta JSON con `Calificación`.

`POST /conclusion-modelo-gpt`

- **Descripción**: Envía datos parciales de análisis y espera una conclusión de seguridad.
- **Entradas**: JSON con `resultado_parcial`.
- **Acciones**: Enviar `POST` con el campo `resultado_parcial`.
- **Resultados Esperados**: JSON con el campo `conclusion`.

4. Pruebas de Carga

Propósito

Evaluar el desempeño de la API bajo distintas cargas de usuario, asegurando que el servicio maneje las solicitudes concurrentes sin fallas.

Herramientas

Locust

Escenarios de Prueba

Escenario de Baja Carga

- **Descripción**: Pruebas con 10 usuarios concurrentes.
- **Resultados Esperados**: Tiempos de respuesta consistentes (promedio < 500 ms).

Escenario de Carga Promedio

- **Descripción**: 50 usuarios concurrentes.
- **Resultados Esperados**: La API mantiene un rendimiento adecuado, con tiempos de respuesta promedio de < 1000 ms.

Escenario de Carga Máxima

- **Descripción**: 100 usuarios concurrentes.
- **Resultados Esperados**: La API responde en menos de 2 segundos con un 95% de éxito.

Métricas

- **Tiempo de Respuesta**: Promedio y máximo.
- **Tasa de Éxito**: % de solicitudes exitosas.
- **Consumo de Recursos**: Uso de CPU y memoria.

Criterios de Aceptación

- La API debe responder en < 2 segundos bajo máxima carga, manteniendo la tasa de éxito en al menos 95%.

5. Pruebas de API

Propósito

Validar la funcionalidad y manejo de errores en los endpoints de la API.

Herramientas

`Pytest`, `Mocking`

Especificación de las Pruebas de API

Endpoint `/consultar-modelo-gpt`

- **Entradas**: `mensaje`
- **Resultados Esperados**: JSON con `Calificación`.

Endpoint `/conclusion-modelo-gpt`

- **Entradas**: `resultado_parcial`
- **Resultados Esperados**: JSON con `conclusion`.

Casos de Prueba

`POST /consultar-modelo-gpt`

- **Identificador**: `API001`
- **Descripción**: Evaluar mensaje en busca de phishing.
- **Datos de Entrada**: `mensaje` con texto de prueba.
- **Acciones**: Enviar `POST` con `mensaje`.
- **Resultados Esperados**: `Calificación`.

8. Anexos

Archivos de Configuración de Pruebas

- Scripts y configuraciones de `Locust`, `Unittest` y `Pytest`.

Logs de Ejecución

- Se incluirán registros y capturas de pantalla como evidencia de los resultados.

Documentación de Herramientas

- Manual de uso y configuración de `Locust`, `Pytest`, `Mocking`