Pruebas de Software Servicio Twitter – X SmishGuard

1. Introducción

Propósito

Este documento especifica las pruebas para validar el servicio Flask que interactúa con la API de Twitter, detectando y publicando alertas sobre mensajes de smishing. Las pruebas cubren el funcionamiento de los endpoints principales, la verificación de bancos colombianos en los mensajes SMS, y aseguran la robustez del servicio al gestionar excepciones y entradas incorrectas.

Alcance de las pruebas

El documento abarca las pruebas de los siguientes endpoints y funciones:

- **Pruebas Unitarias**: Funciones principales y validación del endpoint `/ping`.
- **Pruebas de API**: Endpoint `/tweet` para publicación en Twitter y verificación de bancos en mensajes SMS.

Objetivos

1. Validar que los endpoints y la función de verificación de bancos funcionan correctamente.
2. Asegurar que el servicio maneje correctamente los errores y las entradas faltantes.
3. Simular el proceso de publicación en Twitter para evitar consumo real de la API.

2. Metodología de Pruebas

Tipos de pruebas

- **Pruebas Unitarias**: Verificación del correcto funcionamiento de las funciones y endpoints.
- **Pruebas de API**: Pruebas del comportamiento del endpoint `/tweet` con distintas entradas.

Herramientas de prueba

- **Unittest** y **Pytest** para pruebas unitarias y de API.
- **Mocking** para simular la interacción con la API de Twitter.

Criterios de aceptación

- Cada endpoint debe responder con el código de estado adecuado (200, 400, 500).
- La función de verificación de bancos debe identificar correctamente las menciones de bancos colombianos en los mensajes SMS.
- La API debe manejar errores y entradas faltantes de manera adecuada.

3. Pruebas Unitarias

Propósito

Validar la funcionalidad individual de los endpoints y la función `verificar_bancos` para asegurar que cada parte del servicio funcione según lo esperado.

Herramientas

`Unittest`

Estrategia de Pruebas

Se utilizará un cliente de pruebas de Flask para simular peticiones a los endpoints. Además, se emplearán mocks para las pruebas que requieren comunicación con Twitter, evitando el consumo de la API real.

Cobertura de Pruebas

Se espera una cobertura del 100% en los endpoints y en la función `verificar_bancos`.

Criterios de Ejecución

Cada prueba unitaria se ejecutará en un entorno de pruebas con el cliente de Flask.

Casos de Prueba

`GET /ping`

- **Descripción**: Verifica que el endpoint responda adecuadamente.
- **Entradas**: Ninguna.
- **Acciones**: Realizar una petición `GET /ping`.
- **Resultados Esperados**: Código de estado 200 y respuesta `{"message": "pong"}`.

`POST /tweet` - Publicación de Tweet

- **Descripción**: Verifica la publicación de un tweet a partir de un mensaje SMS.
- **Entradas**: JSON con el campo `sms`.
- **Acciones**: Realizar un `POST /tweet` con un mensaje SMS que mencione un banco.
- **Resultados Esperados**: Respuesta con código 200, incluyendo un `id` del tweet simulado y el texto del tweet.

`POST /tweet` - Falta de Campo `sms`

- **Descripción**: Verifica que el endpoint devuelva un error al faltar el campo `sms`.
- **Entradas**: JSON vacío.
- **Acciones**: Realizar un `POST /tweet` sin el campo `sms`.
- **Resultados Esperados**: Respuesta con código 400 y mensaje `{'error': 'No se proporcionó ningún mensaje SMS'}`.

`verificar_bancos`

- **Descripción**: Verifica que la función detecte correctamente las menciones de bancos en los mensajes.
- **Entradas**: Mensaje SMS con el nombre de un banco.
- **Acciones**: Llamar a `verificar_bancos` con distintos mensajes.
- **Resultados Esperados**: La función retorna las etiquetas de bancos correspondientes.

4. Pruebas de API

Propósito

Validar la interacción y el manejo de errores en los endpoints de la API.

Herramientas

Pytest y **Mocking**

Especificación de las Pruebas de API

Endpoint `/ping`

- **Entradas**: Ninguna.
- **Acciones**: Realizar una petición `GET /ping`.
- **Resultados Esperados**: Código de estado 200 y JSON `{"message": "pong"}`.

Endpoint `/tweet`

- **Entradas**: `sms` con un mensaje de prueba.
- **Acciones**: Realizar una petición `POST /tweet` con el mensaje.
- **Resultados Esperados**: Código de estado 200, incluyendo `id` y `text` del tweet.

Casos de Prueba

`POST /tweet` - Tweet Valido

- **Identificador**: `API001`
- **Descripción**: Envía un mensaje SMS válido y simula la respuesta de la API de Twitter.
- **Datos de Entrada**: JSON con `sms`.
- **Acciones**: Enviar `POST /tweet`.
- **Resultados Esperados**: Respuesta con `id` y `text` del tweet.

`POST /tweet` - Faltante de `sms`

- **Identificador**: `API002`
- **Descripción**: Envía una petición sin el campo `sms`.
- **Datos de Entrada**: JSON vacío.
- **Acciones**: Enviar `POST /tweet`.
- **Resultados Esperados**: Código de estado 400 y mensaje `{'error': 'No se proporcionó ningún mensaje SMS'}`.