## **1. Objetivo y Funcionalidad General**

El propósito de este código es crear un bot para Slack que, utilizando **Socket Mode** y la **Slack SDK para Python**, escuche los mensajes en tiempo real. Cuando el bot detecta que un mensaje contiene alguna de las palabras clave definidas (por ejemplo, "ayuda", "it", "no sirve", etc.), responde enviando un mensaje efímero al usuario que envió el mensaje. Este mensaje efímero es visible únicamente para el usuario que lo ha publicado, y en él se le informa que su solicitud de ayuda será revisada.

## **2. Requisitos Previos**

* **Slack App Configurada**: Se debe tener una aplicación de Slack con:  
  + Un **Bot Token** (xoxb-...) con los permisos necesarios, tales como chat:write, channels:read, etc.
  + Un **App-Level Token** (xapp-...) habilitado para Socket Mode con el scope connections:write.

**Biblioteca slack\_sdk**: Instalar mediante:  
  
 bash  
Copiar  
pip install slack\_sdk

* **Permisos y Event Subscriptions**: La app debe estar suscrita a eventos de mensajes (por ejemplo, message.channels o message.groups) y el bot debe estar invitado a los canales donde se espera responder.

## **3. Descripción del Código**

### **Importación de Módulos**

python

Copiar

import time

from slack\_sdk import WebClient

from slack\_sdk.socket\_mode import SocketModeClient

from slack\_sdk.socket\_mode.request import SocketModeRequest

from slack\_sdk.socket\_mode.response import SocketModeResponse

* **time**: Se utiliza para mantener el proceso activo en un bucle infinito.
* **WebClient**: Permite interactuar con la API web de Slack (por ejemplo, enviar mensajes).
* **SocketModeClient**: Cliente que se conecta a Slack mediante Socket Mode para recibir eventos en tiempo real.
* **SocketModeRequest** y **SocketModeResponse**: Permiten procesar y responder a los eventos recibidos.

### **Configuración de Tokens y Palabras Clave**

python

Copiar

# Token de Bot de Slack (xoxb-...)

SLACK\_BOT\_TOKEN =

# Token de nivel de aplicación para Socket Mode (xapp-...)

SLACK\_APP\_TOKEN =

# Lista de palabras clave que disparan la respuesta

KEYWORDS = ["ayuda", "it", "no sirve", "no funciona", "no conecta", "no abre"]

* **SLACK\_BOT\_TOKEN**: Es el token que identifica a la aplicación y permite ejecutar acciones como enviar mensajes.
* **SLACK\_APP\_TOKEN**: Es el token utilizado para conectar en **Socket Mode**. Debe tener el scope connections:write.
* **KEYWORDS**: Lista de términos que, al ser detectados en el mensaje, disparan la respuesta del bot.

### **Inicialización de Clientes de Slack**

python

Copiar

# Cliente web de Slack (para enviar mensajes)

web\_client = WebClient(token=SLACK\_BOT\_TOKEN)

# Cliente de Socket Mode (para recibir eventos en tiempo real)

socket\_client = SocketModeClient(

app\_token=SLACK\_APP\_TOKEN,

web\_client=web\_client

)

* **web\_client**: Se usa para realizar llamadas a la API web de Slack (por ejemplo, chat\_postEphemeral para enviar mensajes efímeros).
* **socket\_client**: Se configura con el token de la app y el cliente web. Este cliente se conecta a Slack vía Socket Mode y recibe eventos.

### **Función de Procesamiento de Eventos**

python

Copiar

def process(client: SocketModeClient, req: SocketModeRequest):

# Procesa solo eventos de tipo "events\_api"

if req.type == "events\_api":

event = req.payload.get("event", {})

# Verifica que sea un mensaje y que no sea un mensaje con subtype (para filtrar eventos automáticos)

if event.get("type") == "message" and "subtype" not in event:

text = event.get("text", "").lower()

channel\_id = event.get("channel", "")

user\_id = event.get("user", "")

# Si el mensaje contiene alguna de las palabras clave, se envía un mensaje efímero al usuario

if any(keyword in text for keyword in KEYWORDS):

web\_client.chat\_postEphemeral(

channel=channel\_id,

text="¡Hola! Vemos que necesitas ayuda. En breve revisaremos tu caso.",

user=user\_id

)

# Envía la respuesta para confirmar que se procesó el evento

response = SocketModeResponse(envelope\_id=req.envelope\_id)

client.send\_socket\_mode\_response(response)

#### **Detalle de la función:**

* **Entrada**: La función process recibe dos argumentos:  
  + client: el objeto SocketModeClient.
  + req: el objeto SocketModeRequest, que contiene el evento recibido desde Slack.
* **Filtrado de Eventos**:  
  + Se verifica que el evento sea de tipo events\_api y específicamente un evento de tipo message.
  + Se ignoran mensajes con un campo subtype para evitar procesar mensajes automáticos o modificados por Slack.
* **Lógica de Respuesta**:  
  + Convierte el texto del mensaje a minúsculas para hacer una comparación insensible a mayúsculas.
  + Usa la función any() para determinar si alguna de las palabras clave definidas en KEYWORDS está presente en el mensaje.
  + Si se cumple, se envía un mensaje efímero mediante chat\_postEphemeral, de modo que solo el usuario que envió el mensaje vea la respuesta.
* **Confirmación del Evento**:  
  + Se crea un objeto SocketModeResponse utilizando el envelope\_id del evento y se envía de vuelta a Slack para confirmar que el evento fue procesado correctamente.

### **Bloque Principal del Programa**

python

Copiar

if \_\_name\_\_ == "\_\_main\_\_":

# Agrega el listener para manejar los eventos

socket\_client.socket\_mode\_request\_listeners.append(process)

# Conecta vía Socket Mode

socket\_client.connect()

print("Bot iniciado y escuchando mensajes (Socket Mode)...")

# Mantén el script en ejecución

while True:

time.sleep(1)

* **Registro del Listener**:  
  + Se agrega la función process a la lista de listeners de socket\_client. Esto indica que cada evento recibido será procesado por dicha función.
* **Conexión en Socket Mode**:  
  + El método connect() establece la conexión con Slack usando el App-Level Token.
* **Bucle Infinito**:  
  + Un bucle while True con time.sleep(1) mantiene el script en ejecución, de modo que el bot continúe recibiendo y procesando eventos en tiempo real.

## **4. Fin y Propósito del Código**

El código tiene como finalidad:

* **Monitorear en tiempo real** los mensajes que se envían en los canales de Slack donde el bot está presente.
* **Detectar ciertos términos de ayuda** (por ejemplo, "ayuda", "no funciona", etc.) que indican que un usuario podría necesitar soporte.
* **Responder de forma privada** (mensaje efímero) al usuario, informándole que su solicitud de ayuda está siendo atendida.
* **Integrar la aplicación con Slack** usando Socket Mode, lo que permite recibir eventos sin tener que exponer un endpoint público.

Este enfoque es muy útil en escenarios donde deseas automatizar respuestas de soporte o activar flujos de trabajo sin inundar el canal con mensajes públicos, manteniendo la comunicación directa y privada con el usuario que lo solicita.  
  
Además de próximamente agregar desiciones e identificar problemas y que el chat vaya brindando una guia al usuario para que este lo puede solucionar, en caso de ser muy grave que el chat lo redirige con nosotros.