Appli desktop de Chat avec CustomTkinter

CustomTkinter est un framework basé sur **Tkinter**, qui permet de créer des interfaces graphiques modernes et stylisées en utilisant des composants personnalisés (widgets). Il ajoute des fonctionnalités telles que les thèmes sombres et clairs, ainsi qu'un style plus moderne pour les applications de bureau.

Voici comment vous pouvez adapter l'application de chat précédente pour utiliser **CustomTkinter** à la place de **PyQt5** :

Installer CustomTkinter

Installez `customtkinter` :

```
pip install customtkinter
```

Créer l'application avec CustomTkinter

Voici un exemple pour créer une application de chat utilisant **Flask-SocketIO** et **CustomTkinter**.

```
# desktop_chat_app_customtkinter.py

import sys
import threading
import asyncio
import socketio
import customtkinter

# Configuration de l'application CustomTkinter
class ChatClient(customtkinter.CTk):
    def __init__(self):
        super().__init__()
        self.title("Flask-SocketIO Chat Client")
        self.geometry("400x300")

        # Configuration du layout
        self.chat_display = customtkinter.CTkTextbox(self, width=380,
height=200)
        self.chat_display.pack(padx=10, pady=10)

        self.input_field = customtkinter.CTkEntry(self, width=280)
        self.input_field.pack(side="left", padx=(10, 0), pady=10)

        self.send_button = customtkinter.CTkButton(self, text="Send",
command=self.send_message)
        self.send_button.pack(side="right", padx=(0, 10), pady=10)
```

```
        # Connexions des événements
        self.input_field.bind("<Return>", lambda event:
self.send_message())

        # Configuration du client SocketIO
        self.sio = socketio.Client()
        self.sio.on('message', self.receive_message)

        # Connexion au serveur
        self.connect_to_server()

    def connect_to_server(self):
        try:
            self.sio.connect('http://127.0.0.1:5000')
            self.chat_display.insert("end", "Connected to the
server.\n")
        except Exception as e:
            self.chat_display.insert("end", f"Connection error: {e}\n")

    def receive_message(self, msg):
        self.chat_display.insert("end", f"Server: {msg}\n")
        self.chat_display.yview("end")  # Scroll vers le bas

    def send_message(self):
        message = self.input_field.get()
        if message:
            self.sio.send(message)
            self.chat_display.insert("end", f"You: {message}\n")
            self.chat_display.yview("end")
            self.input_field.delete(0, "end")

    def close_event(self):
        # Ferme proprement la connexion SocketIO
        if self.sio.connected:
            self.sio.disconnect()
        self.destroy()

# Lancer l'application de bureau CustomTkinter
def main():
    app = ChatClient()
    app.protocol("WM_DELETE_WINDOW", app.close_event)
    app.mainloop()

if __name__ == "__main__":
    main()
```

PROF

Configuration du serveur Flask

Assurez-vous que le serveur Flask tourne comme décrit dans l'exemple précédent.

Exécuter l'application

Lancez l'application de bureau avec la commande :

```
python3 desktop_chat_app_customtkinter.py
```

Explications

1. **CustomTkinter** : Les widgets de base comme `CTk`, `CTkTextbox`, `CTkEntry`, et `CTkButton` sont utilisés pour créer une interface moderne.
2. **SocketIO** : Le client se connecte au serveur Flask via SocketIO pour envoyer et recevoir des messages.
3. **Gestion des événements** : Les événements `Return` (entrée) et les clics sur le bouton d'envoi sont gérés pour envoyer les messages au serveur.

Suggestions :

a. Ajouter une option de changement de thème (sombre/clair) dans l'application CustomTkinter.
b. Utiliser des threads pour gérer les communications réseau pour une interface plus réactive.