Gemini MCP Chatbot Frontend

Questo progetto fornisce un'interfaccia web per il tuo chatbot basato sul Model Control Protocol (MCP) che utilizza il modello Gemini di Google.

Caratteristiche

- Interfaccia web responsiva realizzata con Flask e Socket.IO
- Supporto per chat in tempo reale
- Visualizzazione formattata dei messaggi con supporto Markdown
- Sistema di log per il monitoraggio delle chiamate ai tool
- Gestione delle sessioni multiple
- Reset della conversazione

Requisiti

- Python 3.7+
- Flask
- Flask-SocketIO
- python-dotenv
- Google Generative AI Python SDK
- File main.py esistente con la classe MCPClient

Installazione

1. Assicurati di avere installato tutti i pacchetti necessari:

```
pip install flask flask-socketio python-dotenv google-generativeai
```

2. Crea un file .env nella directory principale del progetto:

```
GOOGLE_API_KEY=la_tua_chiave_api_gemini
SECRET_KEY=una_chiave_segreta_per_flask
```

3. Crea una directory templates e inserisci il file index.html fornito.

4. Assicurati che il file main.py con la classe MCPClient sia nella stessa directory di app.py.

Struttura del progetto

```
/progetto
  ├── main.py        # File esistente con la classe MCPClient
  ├── app.py         # Server Flask + Socket.IO
  ├── .env           # Variabili d'ambiente
  └── /templates
      └── index.html    # Interfaccia utente web
```

Utilizzo

1. Avvia il server Flask:

```
python app.py
```

2. Apri un browser web e visita http://localhost:5000.
3. Nella pagina web:
    a. Inserisci il percorso al tuo script server MCP (ad esempio, server.py)
    b. Clicca su "Inizializza" per avviare il client MCP
    c. Una volta inizializzato, puoi inviare messaggi e ricevere risposte
    d. Usa il pulsante "Reset Chat" per iniziare una nuova conversazione
    e. Visualizza i log di sistema per monitorare le chiamate ai tool

Come funziona

Il frontend utilizza Flask come server web e Socket.IO per la comunicazione in tempo reale. Quando un utente si connette, viene creata una nuova istanza di MCPClient dedicata alla sessione. I messaggi dell'utente vengono inviati al client MCP che utilizza Gemini per generare risposte e chiamare strumenti quando necessario.

Il flusso di comunicazione è il seguente:

1. L'utente si connette al server web
2. L'utente inizializza il client MCP con lo script del server

3. L'utente invia un messaggio
4. Il server elabora il messaggio tramite il client MCP
5. Le risposte e i log vengono inviati all'utente in tempo reale

Personalizzazione

- Modifica templates/index.html per cambiare l'aspetto dell'interfaccia
- Aggiorna app.py per aggiungere nuove funzionalità o modificare il comportamento esistente
- Aggiungi nuove route Flask se necessario

Risoluzione dei problemi

- Se riscontri errori di connessione, verifica che lo script server MCP sia corretto e accessibile
- Controlla che la chiave API di Google sia valida
- Verifica che tutti i pacchetti Python richiesti siano installati
- I log di sistema possono aiutare a identificare problemi con le chiamate ai tool

Note importanti

- Questo frontend è progettato per funzionare con la classe MCPClient fornita nel file main.py
- Assicurati che il tuo script server MCP sia compatibile con il client MCP
- La gestione degli errori è implementata per catturare e visualizzare i problemi più comuni