Web-app in Python per interagire con diversi LLM e per l'analisi di prompt.

Ad oggi i modelli supportati sono:

- GPT-3.5-turbo
- Google Gemini
- Phi3
- Gemma2
- Llama3.1
- Mistral-nemo
- Claude3.5
- Qwen2

Come anticipato in precedenza si tratta di una Web-App deployata su Microsoft Azure al fine di renderla accessibile a chiunque tramite il web.

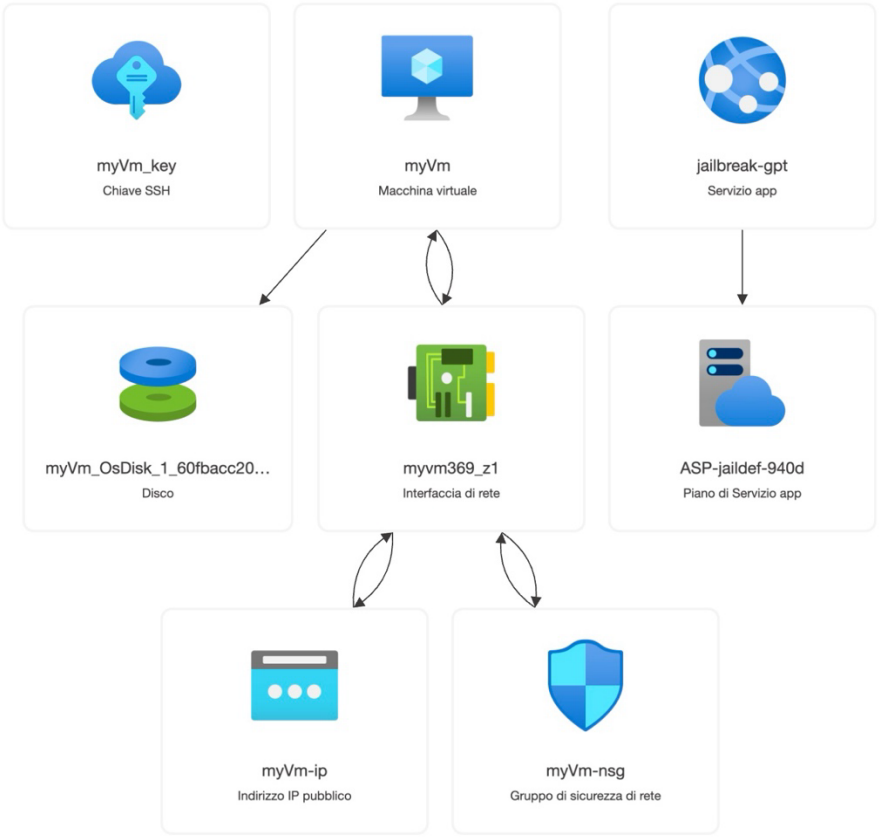
E' raggiungibile tramite l'URL: https://jailbreak-gpt.azurewebsites.net/

Per quanto riguarda i modelli questi sono eseguiti tramite Ollama, un framework che permette l'esecuzione di LLM opensource in maniera rapida ed efficiente in locale, mentre altri permettevano l'utilizzo direttamente tramite le API proprietarie.

La webapp è stata progettata e realizzata interamente in Python insieme a Streamlit, un potente framework open-source utilizzato da data scientists e ingegneri AI/ML per realizzare app del genere.

DEPLOY SU AZURE E VM

Come accennato in precedenza Ollama è uno strumento che permette di eseguire vari LLM in locale. Per via delle risorse necessarie si è pensato di hostare e gestire Ollama direttamente su VM nel cloud.



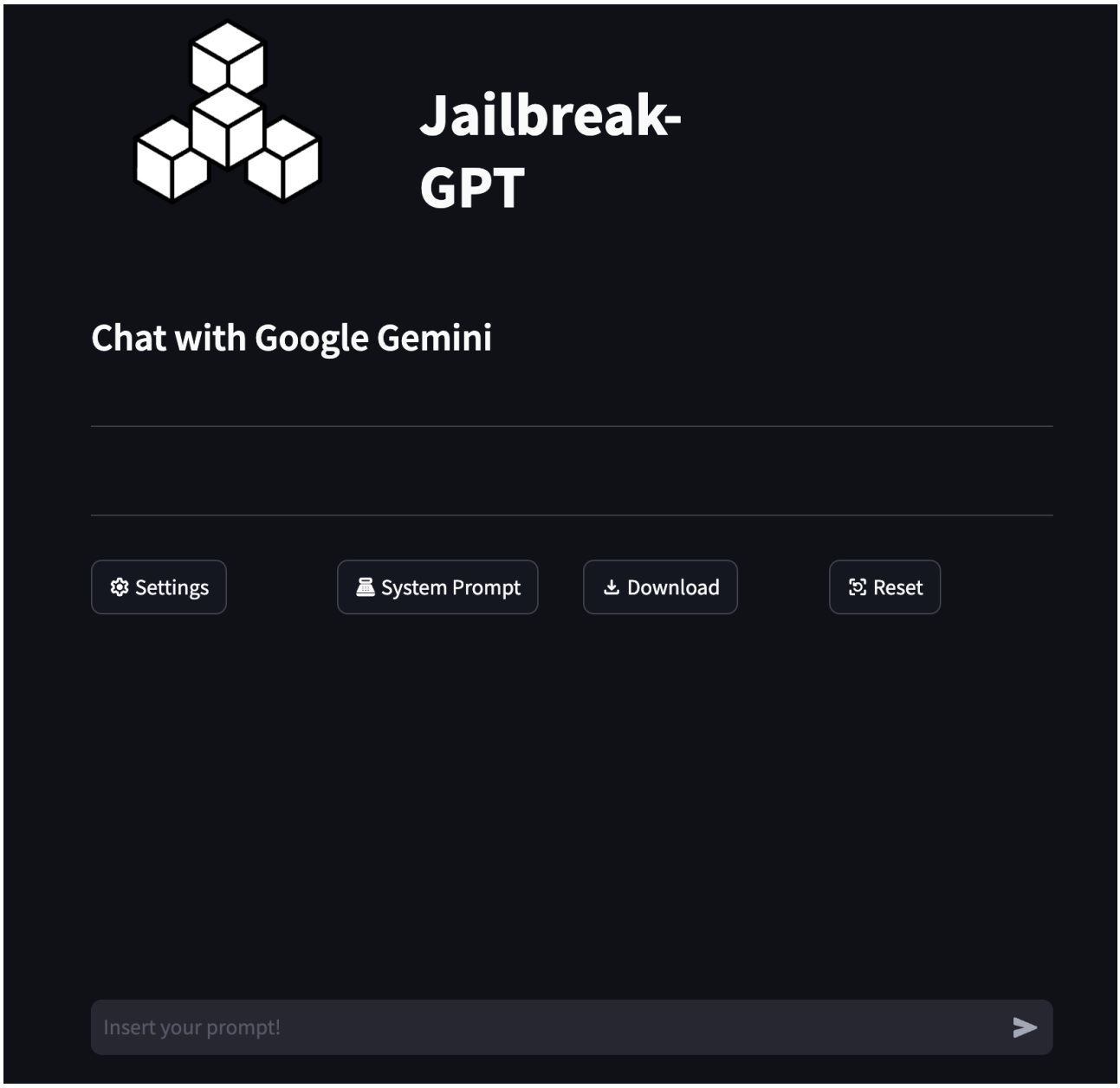
Img dell'architettura

Dunque, una volta istanziata la VM Ubuntu vi è stato installato "Docker" al suo interno e tramite quest'ultimo è stato fatto il pull di Ollama. A questo punto è stato tirato su un container con Ollama, che comunicasse con l'esterno tramite la porta 11434 (+ opportune modifiche al firewall della VM Azure), e vi è stato fatto il pull dei vari LLM.

Una volta configurato opportunamente Ollama, è stato pushato tutto su GitHub e tirata su l'istanza della Web-App.

(Per motivi di costi la VM è spenta e accesa solo nel momento di utilizzo effettivo).

MODALITA’ DI UTILIZZO



Una volta selezionato dal menu il modello con il quale si vuole interagire ci si trova dinanzi questa schermata, la quale permette di iniziare una conversazione con il modello scelto.

Inoltre vi sono alcuni bottoni che permettono:

- Settings: permette di modificare alcuni parametri del LLM utilizzato quali temperature, top_p, top_k e max_output_token (**)
- System Prompt: permette di inserire un prompt di sistema (*)
- Download: permette di esportare la chat con il modello in un file .JSON
- Reset: reset del modello

(*) Un "system prompt" per un Language Model (LLM) è un tipo di prompt utilizzato per configurare o guidare il comportamento del modello durante una sessione di conversazione o generazione di testo. A differenza dei prompt standard che possono semplicemente essere domande o frasi che l'utente vuole completare, i system prompt sono progettati per stabilire il contesto, impostare le regole di comportamento, o fornire istruzioni specifiche al modello.

(**) Ecco una breve descrizione dei parametri per un LLM:

1. **Temperature**: Controlla la casualità delle risposte. Valori bassi (es. 0.2) rendono le risposte più deterministiche, mentre valori alti (es. 1.0) le rendono più varie e creative.

2. **Top_p (nucleus sampling)**: Seleziona i token dalla distribuzione delle probabilità fino a quando la somma delle probabilità raggiunge una certa soglia \(p\). Valori bassi rendono la selezione più ristretta e sicura.

3. **Top_k**: Limita la selezione ai token con le \(k\) probabilità più alte. Se impostato a 50, il modello considera solo i 50 token più probabili.

4. **Max_output_tokens**: Imposta il numero massimo di token che il modello può generare in una risposta.

TOKEN_COUNTS

E’ stata inserita questa funzione per una stima dei costi e delle risorse. Un token può essere una parola intera, una parte di parola, un carattere singolo o persino un simbolo. La “tokenizzazione” è il processo di suddivisione del testo in queste unità. I token servono come input per i modelli di linguaggio, che li utilizzano per prevedere e generare nuovo testo.

TODO:

- Revisione e pulizia del codice
- Payments delle API di Claude e OpenAI
-