Capitolo 8: Il sistema di Function Calling di OpenAI

Questo capitolo esplora in dettaglio le funzionalità operative implementate nel chatbot AI, con particolare enfasi sul meccanismo di function calling che consente all'assistente OpenAI di eseguire operazioni concrete sulla piattaforma proprietaria Konsolex. Questa caratteristica rappresenta il ponte tra le capacità di comprensione del linguaggio naturale dell'AI e l'esecuzione di azioni tecniche specifiche, permettendo al sistema di offrire un supporto operativo completo oltre alla semplice assistenza informativa.

8.1 Architettura del Function Calling

Il function calling rappresenta uno degli aspetti più innovativi del sistema, consentendo all'AI di riconoscere quando è necessario eseguire operazioni tecniche specifiche e invocarle in modo trasparente all'interno del flusso conversazionale.

8.1.1 Principi di Funzionamento

Il meccanismo di function calling si basa sulla capacità dell'API OpenAI di analizzare il contesto della conversazione, determinare quando è necessaria l'esecuzione di una funzione specifica, e richiedere parametri appropriati. L'architettura implementata nel file `openai-handlers.ts` segue un flusso ben definito:

1. L'utente esprime una richiesta che richiede un'azione tecnica
2. L'AI analizza la richiesta e determina quale funzione è necessaria
3. Il sistema raccoglie i parametri richiesti, chiedendo informazioni aggiuntive all'utente se necessario
4. La funzione viene eseguita tramite l'Endpoint Handler (componente che gestisce le comunicazioni con le API esterne)
5. Il risultato dell'operazione viene restituito all'AI che lo incorpora nella risposta all'utente

8.1.2 Gestione del Ciclo di Richiesta-Esecuzione

La gestione del ciclo di vita completo di una richiesta di funzione è implementata principalmente nella funzione `handleToolCalls` del file `openai-handlers.ts`, che si occupa di:

1. Ricevere le richieste di tool (function call) dall'API OpenAI
2. Estrarre il nome della funzione e i parametri dalla richiesta
3. Invocare la funzione appropriata tramite il dispatcher `executeTool`
4. Gestire eventuali errori durante l'esecuzione
5. Restituire i risultati all'API OpenAI per incorporarli nella risposta

Questa implementazione consente un'integrazione fluida delle operazioni tecniche all'interno del flusso conversazionale, mantenendo l'esperienza utente coerente e naturale.

8.1.3 Dichiarazione e Definizione delle Funzioni

Le funzioni disponibili per l'AI sono definite come "tools" nel file `openai-tool.ts`, utilizzando un formato JSON standardizzato secondo la struttura richiesta dall'API OpenAI:

```
// Esempio semplificato di definizione di una funzione
{
  "type": "function",
  "function": {
    "name": "restartServer",
    "description": "Riavvia un server specificato",
    "strict": true,
    "parameters": {
      "type": "object",
      "properties": {
        "serverName": {
          "type": "string",
          "description": "Nome del server da riavviare"
        }
      },
      "required": ["serverName"],
      "additionalProperties": false
    }
  }
}
```

Questa struttura dichiarativa specifica:

- Nome della funzione
- Descrizione del suo scopo
- Parametri richiesti con tipi e descrizioni
- Requisiti sui parametri (quali sono obbligatori)

Questo approccio permette all'API OpenAI di comprendere:

- Quale funzione può risolvere un determinato problema
- Quali parametri sono necessari per l'esecuzione
- Come guidare l'utente nella fornitura dei parametri mancanti

8.2 Implementazione Tecnica delle Function Calls

L'implementazione tecnica del sistema di function calling rappresenta uno degli aspetti più sofisticati dell'architettura, integrando le capacità di OpenAI con le API della piattaforma propietaria Konsolex.

8.2.1 Gestione dell'Esecuzione

Il cuore dell'implementazione è rappresentato dalla funzione `executeTool` nel file `openai-handlers.ts`, che funge da dispatcher per le varie funzioni disponibili:

```
// Implementazione semplificata del dispatcher
async function executeTool(functionName: string, args: any, userId: string, msgId:
number): Promise<any> {
  switch (functionName) {
    case 'restartServer':
      return await restartServer(userId, args.serverName);
```

```
      case 'restartMysql':
        return await restartMysqlOrPostfix(userId, args.serverName, "mysql");
      case 'getDomainList':
        return await getDomainList(userId);
      // Altri casi...
      default:
        throw new Error(`Function ${functionName} not implemented`);
    }
  }
```

Questa implementazione centralizzata facilita:

- L'aggiunta di nuove funzioni
- La gestione uniforme degli errori
- Il monitoring delle chiamate di funzione
- L'implementazione di controlli di sicurezza centralizzati

8.2.2 Integrazione con il Flusso Conversazionale

L'integrazione delle function calls nel flusso conversazionale avviene attraverso la funzione `handleToolCalls`, che viene invocata quando l'API OpenAI richiede l'esecuzione di una funzione:

```
// Implementazione semplificata del gestore di tool calls
async function handleToolCalls(openai: OpenAI, run: any, threadId: string, runId:
string, userId: string, msg: Message) {
  if (run.required_action?.type === 'submit_tool_outputs') {
    const toolCalls = run.required_action.submit_tool_outputs.tool_calls;
    const toolOutputs = [];

    for (const toolCall of toolCalls) {
      const functionName = toolCall.function.name;
      const functionArgs = JSON.parse(toolCall.function.arguments);

      try {
        const functionResult = await executeTool(functionName, functionArgs,
userId, msg.id!);
        toolOutputs.push({
          tool_call_id: toolCall.id,
          output: typeof functionResult === 'string' ? functionResult :
JSON.stringify(functionResult)
        });
      } catch (error) {
        // Gestione errori...
      }
    }

    // Sottomissione dei risultati all'API OpenAI
    if (toolOutputs.length > 0) {
      await openai.beta.threads.runs.submitToolOutputs(threadId, runId, {
tool_outputs: toolOutputs });
    }
```

```
    }
  }
```

Questa implementazione garantisce che:

1. Le richieste di function calling vengano correttamente interpretate
2. I risultati delle funzioni vengano restituiti all'API OpenAI
3. Gli errori vengano gestiti in modo appropriato
4. Il flusso conversazionale rimanga coerente anche in caso di operazioni tecniche complesse

8.3 Estensibilità del Sistema di Funzioni

Una caratteristica fondamentale del sistema è la sua estensibilità, progettata per facilitare l'aggiunta di nuove funzionalità operative senza modifiche significative al core dell'architettura.

8.3.1 Aggiunta di Nuove Funzioni

Il processo di aggiunta di nuove funzioni è stato standardizzato per garantire semplicità e coerenza:

1. Definizione della funzione in formato JSON secondo lo schema OpenAI
2. Implementazione della funzione effettiva nell'Endpoint Handler
3. Aggiunta del caso corrispondente nel dispatcher `executeTool`
4. Aggiornamento della documentazione interna

8.3.2 Versioning delle Funzioni

Il sistema implementa un approccio di versioning implicito delle funzioni, permettendo:

1. L'evoluzione delle interfacce di funzione nel tempo
2. Il mantenimento della compatibilità con conversazioni esistenti
3. L'introduzione di nuovi parametri o comportamenti in modo non distruttivo

Questo approccio garantisce che il sistema possa evolvere continuamente senza compromettere l'esperienza utente o la coerenza delle conversazioni in corso.

8.4 Sicurezza e Validazione nelle Function Calls

La sicurezza rappresenta un aspetto fondamentale nell'implementazione delle function calls, considerando che queste funzioni eseguono operazioni potenzialmente critiche sui sistemi degli utenti.

8.4.1 Validazione dei Parametri

Il sistema implementa rigorosi controlli di validazione sui parametri ricevuti:

1. Validazione di tipo tramite la definizione JSON Schema
2. Controlli aggiuntivi sui valori ammissibili
3. Sanitizzazione degli input per prevenire iniezioni o altri attacchi
4. Risoluzione dei valori ambigui attraverso matching intelligente (come nella funzione `findBestMatch`)

8.4.2 Autorizzazione delle Operazioni

Ogni richiesta di funzione viene autenticata e autorizzata prima dell'esecuzione:

1. Verifica dell'identità dell'utente tramite l'ID utente
2. Controllo delle autorizzazioni rispetto alla risorsa target
3. Limitazione delle operazioni alle risorse di proprietà dell'utente
4. Logging delle operazioni per audit e tracciabilità

8.5 Panoramica delle Principali Funzionalità Implementate

Il sistema implementa diverse categorie di funzionalità operative che permettono all'utente di interagire con l'infrastruttura tecnologica attraverso semplici richieste in linguaggio naturale:

Operazioni sui Server e Servizi

Il sistema consente agli utenti di eseguire operazioni di gestione sui server come:

- Riavvio di server completi tramite `restartServer`
- Riavvio di servizi specifici come MySQL (sistema di gestione database) o Postfix (server email) tramite `restartMysqlOrPostfix`
- Ottenere elenchi e informazioni sui server disponibili con `getServerList`

Queste funzionalità permettono la risoluzione rapida di problemi comuni senza necessità di accedere a pannelli di controllo tecnici.

Gestione Domini, Email e Container

Il chatbot gestisce anche operazioni relative a domini, configurazioni email e container:

- Verifica disponibilità e gestione domini tramite funzioni come `getDomainList`
- Supporto alla configurazione email con `restartPostfix`
- Gestione dei container con `restartWebContainer` e `restartDbContainer`

Sistema di Escalation

Quando una richiesta supera le capacità del sistema automatizzato, la funzione `openTicket` crea un ticket di supporto, notifica gli amministratori e informa l'utente che la sua richiesta verrà presa in carico da un operatore umano, garantendo continuità nell'assistenza.

Queste funzionalità rappresentano esempi concreti di come il function calling permetta di combinare l'interfaccia conversazionale intuitiva con l'esecuzione di operazioni tecniche complesse, semplificando l'esperienza di gestione dell'infrastruttura per l'utente finale.

8.6 Conclusioni

Il sistema di function calling implementato rappresenta un importante progresso nell'evoluzione del supporto tecnico automatizzato. L'approccio adottato dimostra come l'intelligenza artificiale possa non solo comunicare, ma anche agire attivamente sull'infrastruttura tecnica, superando le limitazioni dei tradizionali sistemi di chatbot puramente informativi.

L'architettura sviluppata si distingue per tre aspetti fondamentali:

1. **Semplificazione dell'esperienza utente**: Gli utenti possono risolvere problemi tecnici complessi esprimendosi in linguaggio naturale, senza necessità di conoscere comandi specifici o interfacce dedicate

2. **Autonomia**: Il sistema può eseguire autonomamente operazioni critiche pur mantenendo rigorosi controlli di sicurezza e validation

3. **Valore economico**: Riduce significativamente il carico di lavoro sul personale tecnico umano automatizzando operazioni ripetitive, con un impatto diretto sulla gestione delle risorse aziendali