



**Kedos**<sup>Srl</sup>  
Information Technology Solutions

# KeGAL

---

RICERCA E SVILUPPO | KEDOS SRL

## Kedos Graph Agents for LLM

VERSIONE 1.0

© Copyright 2025 Kedos S.r.l

Via Chiavari, 5/E - 43125 Parma (PR)  
CF, P.Iva e Iscr. Reg. Imp. Parma  
02629980349, R.E.A. PR-254346

Phone

Milano +39 02 36741057  
Torino +39 011 19466570  
Parma +39 0521 1554221

Online

[info@kedos-srl.it](mailto:info@kedos-srl.it)  
[www.kedos-srl.it](http://www.kedos-srl.it)

---

---

---

# Premessa

---

**Kedos srl** nasce nel 2012 da un team di persone che per oltre dieci anni hanno lavorato insieme nel settore ICT in ambito bancario e che da sempre condividono approcci, metodi e obiettivi.

Le diverse competenze presenti, da quelle applicative a quelle sistemistiche, sono state integrate per la definizione di soluzioni concrete. La società si divide tra le sedi di Milano, di Torino e di Parma.

I servizi proposti includono consulenza nel settore IT e sviluppo di soluzioni software per vari settori aziendali. A partire dal 2020 la società ha attivato un team di Ricerca e Sviluppo su temi di innovazione tecnologica con particolare focus sull'intelligenza artificiale, sui motori 3D per il web e sull'utilizzo di tecnologie blockchain per applicazioni di business.

Per maggiori informazioni sulle attività di Ricerca e Sviluppo è possibile consultare la seguente pagina web: <https://www.kedos-srl.it/ricerca-e-sviluppo/>.

Il documento che state leggendo fa parte di una serie di white paper sulle novità sviluppate dal team di Ricerca e Sviluppo utilizzando un linguaggio più divulgativo che tecnico.

# Indice

---

---

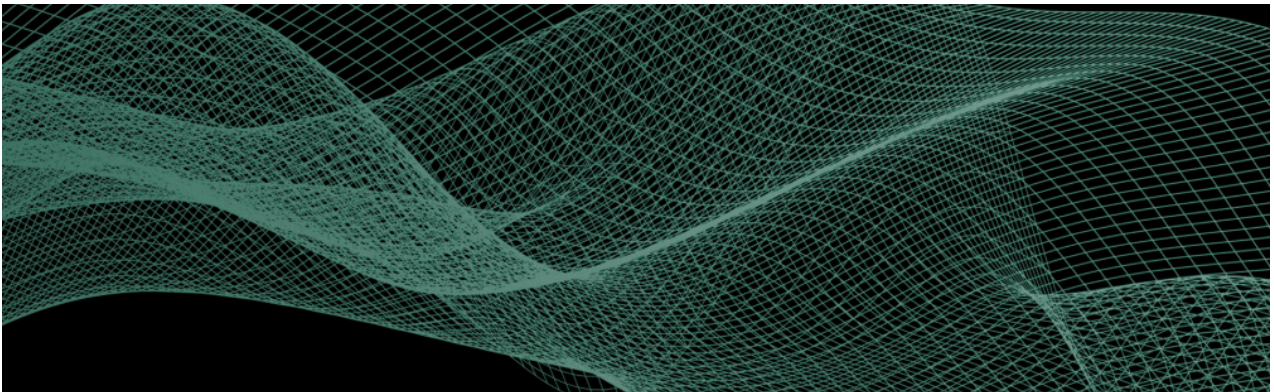
LLM Introduzione	4
Gli agenti Large Language Models	5
KeGAL	7
KeGAL Architettura	9
KeGAL Esempio	11
KeGAL Esempi di utilizzo	12
KeGAL Prompt Guard	13
Caso d'uso KeCoach	15
Caso d'uso KeAssistant	17
Conclusioni	22

---

# LLM

## Introduzione

---



**KeGAL è un framework modulare e scalabile per la costruzione di agenti intelligenti.**

**Negli ultimi anni, i Large Language Models (LLM) hanno rivoluzionato il modo in cui interagiamo con l'intelligenza artificiale, dimostrando straordinarie capacità di generazione e comprensione del linguaggio naturale.**

Tuttavia, questi modelli presentano limitazioni intrinseche, come la dipendenza dai dati su cui sono stati addestrati, la mancanza di memoria a lungo termine e l'incapacità di accedere dinamicamente a informazioni aggiornate o private. Per superare questi limiti, le architetture ad agenti stanno emergendo come una soluzione fondamentale, consentendo l'orchestrazione di modelli LLM in ambienti più complessi e integrati. Un approccio particolarmente efficace è quello basato su Retrieval-Augmented Generation (RAG), che combina la potenza degli LLM con database esterni e fonti di conoscenza aggiornabili, garantendo risposte più accurate, affidabili e contestualizzate. Il progetto Kedos Graph Agents For LLM (KeGAL) nasce con l'obiettivo di fornire un framework modulare e scalabile per la costruzione di agenti intelligenti, ottimizzando l'interazione tra LLM e fonti di dati private o dinamiche.

Il framework è stato rilasciato con licenza open source MIT ed è disponibile all'indirizzo <https://github.com/Kedos-srl/kegal>.

---

LLM

# Gli agenti Large Language Models

---

Gli agenti LLM sono sistemi di intelligenza artificiale che sfruttano i Large Language Models (LLM) con l'aggiunta di strumenti (tools) e memoria per eseguire compiti, prendere decisioni e interagire in modo autonomo con utenti o altri sistemi.

In generale un agente opera prendendo decisioni basate sulle informazioni percepite dall'ambiente, nel caso degli LLM dal contesto, ossia dal prompt in ingresso. Tuttavia, in scenari complessi, un singolo agente spesso non dispone di una conoscenza preesistente sufficiente per risolvere in autonomia tutte le sotto-attività necessarie al raggiungimento di un obiettivo più ampio. Per superare questa limitazione, gli agenti si affidano a strumenti esterni, che possono includere basi di dati strutturate, motori di ricerca, API di terze parti e persino altri agenti specializzati. L'interazione con questi strumenti consente all'agente di recuperare informazioni mancanti e aggiornare il proprio stato di conoscenza. Questo processo iterativo permette una valutazione dinamica del piano d'azione, con meccanismi di autocorrezione e adattamento ad ogni passo dell'esecuzione.

Ad esempio, consideriamo un utente che affida a un agente LLM il compito di ragionare su una procedura interna ad un'azienda per risolvere un problema di business. L'agente, basato su un LLM (Large Language Model), non dispone nativamente della conoscenza di delle procedure interne ad un'azienda, ma può colmare questa lacuna interrogando un database esterno contenente le informazioni aziendali.

Nonostante l'accesso a questi dati, l'agente potrebbe non essere ancora in grado di determinare in modo autonomo quali fonti aziendali consultare, ad esempio un database relazionale, una cartella contenente delle procedure in PDF, un database NoSQL con lo storico dei ticket di assistenza dei clienti. A questo punto, viene generata una sotto-attività, che prevede il coinvolgimento di un agente "di ragionamento" che consente di valutare la migliore strategia di ricerca.

L'agente può fondere i dati acquisiti nel contesto del prompt fornendo una sintesi delle informazioni necessarie per fornire una risposta. Questo paradigma di cooperazione tra agenti e strumenti esterni è ciò che rende le architetture ad agenti più flessibili ed estensibili rispetto ai modelli AI tradizionali, abilitando capacità di ragionamento più avanzate attraverso il retrieval di conoscenza in tempo reale e l'orchestrazione di competenze specializzate.

---

# KeGAL

---

KeGAL è un framework ad agenti per LLM, ossia uno strumento in grado di programmare l'interazione con un Large Language Model tramite l'utilizzo di semplici programmi definiti dall'utente.

In KeGAL, il concetto di agente è più ampio rispetto a quello di un tradizionale tool: un agente descrive un compito specifico all'interno di un contesto di esecuzione chiuso, simile a un blocco di codice eseguibile in un linguaggio di programmazione. In questo, l'approccio di prompt engineering utilizzato in KeGAL è simile a quello di un Linguaggio del primo ordine dove gli enunciati (prompt) seguono conseguenze logiche.

Il comportamento di un agente è stabilito attraverso tecniche di prompt engineering e tramite la gestione delle richieste e delle risposte di un LLM in un linguaggio di programmazione (Python).

Il framework KeGAL è basato sull'interazione di questi agenti secondo un'architettura prestabilita, definita tramite uno schema a grafo. Ogni agente rappresenta un nodo del grafo e ogni interazione è schematizzata attraverso archi. La risposta di un agente rappresenta l'input per un altro agente (Figura 1).

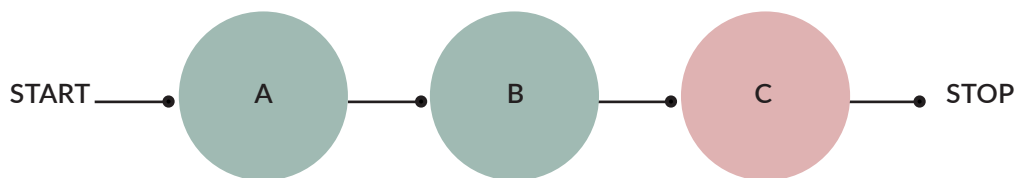


Figura 1: un semplice grafo con 3 agenti: A, B, C

All'interno del framework KeGAL gli agenti vengono definiti attraverso dei file di configurazione in JSON o in YAML e sviluppati tramite delle funzioni in linguaggio Python. Avendo la possibilità di interagire con qualsiasi web API, questi agenti possono richiamare funzionalità esterne sviluppate con qualsiasi altra tecnologia o linguaggio di programmazione, rendendo così l'architettura molto flessibile. Questa architettura a grafo consente di implementare degli agenti generici come ad esempio un agente di guardia (guard node) che controlli

il linguaggio dell'utente, ad esempio evitando l'utilizzo di un linguaggio non appropriato o della scelta dell'argomento di discussione, per tenere la discussione su certi argomenti prestabiliti. Questa architettura a grafo (fig.2) è uno schema generico di un'architettura a grafo con KeGAL.

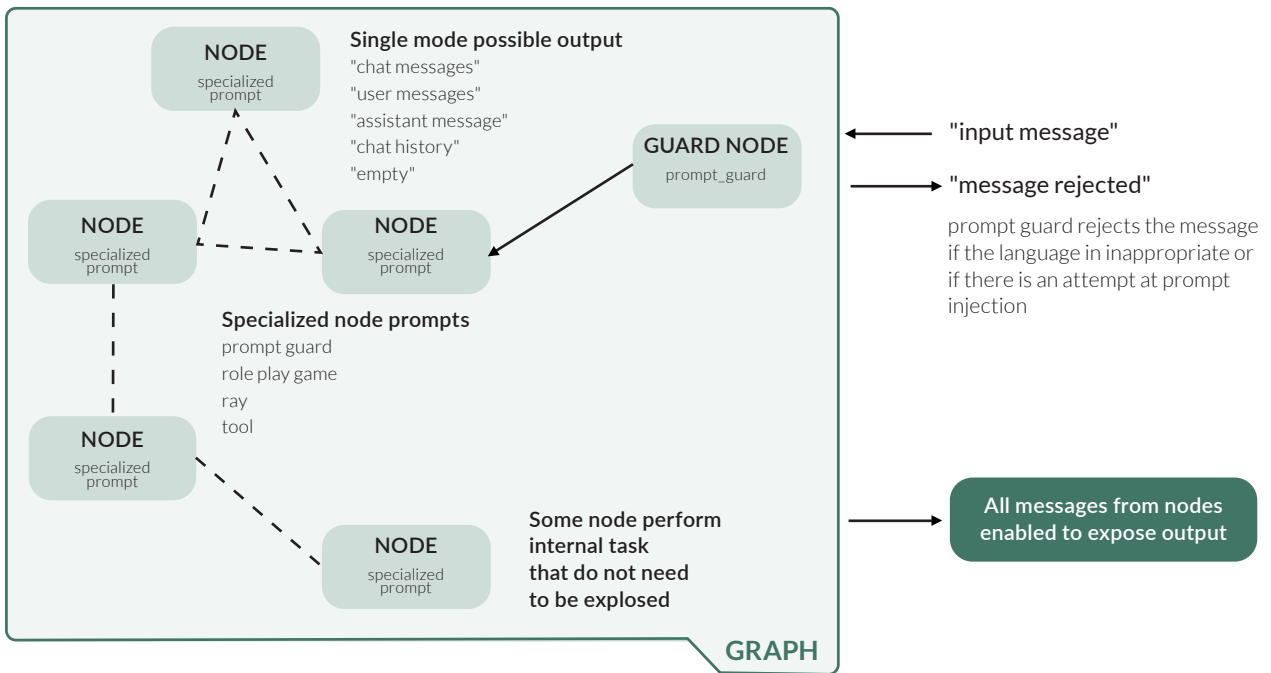


Figura 2: un esempio di architettura ad agenti tramite grafo



# KeGAL

## Architettura

---

Il concetto di un'architettura prompt-centrica per la gestione di processi senza linguaggi di programmazione tradizionali è un approccio innovativo. Estendendo questa idea, possiamo considerare come ogni nodo del grafo possa sfruttare le proprietà emergenti dei nuovi modelli basati sul ragionamento.

In questo sistema, ogni nodo non funziona semplicemente come un punto di elaborazione passivo, ma come un'entità che sfrutta le capacità di ragionamento dei modelli di linguaggio più recenti.

I nodi possono quindi:

- Analizzare autonomamente il contesto e adattare le proprie operazioni
- Formulare ipotesi su come procedere con l'elaborazione basandosi su determinati input
- Valutare criticamente i propri output prima di trasmetterli ad altri nodi
- Risolvere conflitti o ambiguità senza intervento esterno

Questi nodi operano pertanto come entità indipendenti con responsabilità circoscritte, mentre le connessioni tra di essi determinano il percorso del flusso informativo attraverso l'intero sistema.

La configurazione di questo ecosistema computazionale segue un processo metodico che inizia con la definizione dei modelli di elaborazione fondamentali, prosegue con l'implementazione di sistemi per gestire i prompt condivisi, e si completa con la descrizione dettagliata dei singoli nodi e l'articolazione delle connessioni che li uniscono.

Questa fase preparatoria non è semplicemente amministrativa, ma costituisce il fondamento che determina sia la struttura che il comportamento dell'intero sistema.

Durante l'esecuzione, osserviamo un meccanismo rigoroso dove ogni nodo interpreta il proprio prompt specifico, esegue esclusivamente le operazioni per cui è stato progettato, e trasmette i risultati validati ai componenti successivi nella catena computazionale. Questo approccio disciplinato garantisce non solo la correttezza delle operazioni, ma anche una completa tracciabilità di ogni passaggio del processo.

I benefici di questa architettura sono sostanziali e multidimensionali. La modularità intrinseca semplifica notevolmente gli interventi di manutenzione e potenziamento, mentre la flessibilità del sistema permette aggiornamenti rapidi dei comportamenti senza richiedere revisioni strutturali complesse. La trasparenza operativa facilita le attività di debugging e monitoraggio, e la scalabilità dell'architettura consente l'integrazione di nuove funzionalità senza ripercussioni sistemiche. In un panorama tecnologico in rapida evoluzione, questa architettura rappresenta un paradigma progettuale che combina robustezza e adattabilità in un framework coeso e sostenibile.

---

# KeGAL

## Esempio

Un esempio pratico di questo workflow potrebbe includere un sistema di elaborazione testuale strutturato come segue:

**Nodo A** (Sicurezza) → **Nodo B** (RAG) → **Nodo C** (Analisi avanzata) → **Nodo D** (Generazione sommari) → **Nodo C** (Analisi output)

**Nodo A:** Agisce come nodo di guardia, verificando e garantendo la sicurezza degli input

**Nodo B:** Elabora i data provenienti da un database vettoriale

**Nodo C:** Applica modelli linguistici per analisi semantica e sintattica

**Nodo D:** Genera riassunti raffinati basati sugli output elaborati

```
models:
- llm: ollama
  version: "llama3.1:8b-instruct"
  aws_con  fig: null
  host: "http://localhost:11434"
  api_key: null
- llm: ollama
  version: "llama3.3:70-instruct"
  aws_con  fig: null
  host: "http://localhost:11434"
  api_key: null
systems :
- text: "prompt guard"
- text: "prompt vector db"
- text: "prompt text analysis"
- text: "prompt text summary"
nodes :
- id: A
  llm: 0
  temperature: 0.0
  show: true
  prompt:
    system: 0
  placeholders:
    post: ""
    user_role: "User"
- id: B
  llm: 1
  temperature: 0.5
  show: true
  prompt:
    system: 1
  placeholders:
    assistant_role: "Wine Expert"
    citations: ""
    history: ""
    post: ""
    user_role: "Customer"
    user_expertise: "Beginner"
- id: C
  llm: 1
  temperature: 0.5
  show: true
  prompt:
    system: 2
  placeholders:
    assistant_role: "Text analyzer"
    post: ""
- id: D
  llm: 1
  temperature: 0.5
  show: true
  prompt:
    system: 3
  placeholders:
    assistant_role: "Text Synthesizer"
    post: ""
edges:
- source: A
  target: B
  attrib: user_message
- source: B
  target: C
  attrib: assistant_message
- source: C
  target: D
  attrib: assistant_message
- source: D
  target: C
  attrib: assistant_message
```

# KeGAL

## Esempi di utilizzo

---

L'architettura KeGAL dimostra la sua efficacia attraverso la possibilità di creare sofisticati sistemi di controllo degli output basati sulla specializzazione dei nodi, dove ogni componente è ottimizzato per gestire compiti specifici utilizzando modelli linguistici di dimensioni contenute. Questa caratteristica architettonica si rivela particolarmente preziosa nel gestire scenari critici dove la precisione e la coerenza dell'output sono fondamentali.

Ne è un esempio la generazione di output strutturati come nel caso di JSON che devono aderire a schemi specifici. In questo contesto, soprattutto i modelli di dimensioni ridotte potrebbero generare strutture non valide se lasciati operare isolatamente.

L'architettura basata sui grafi di KeGAL permette di creare un sistema di verifiche e correzioni stratificate che garantiscono la conformità dell'output alle specifiche richieste, permettendo anche a modelli molto piccoli di mantenere un'affidabilità comparabile ai grandi modelli (es. llama-3.18b confrontato con gpt-4o).

---

# Esempi di utilizzo

## Prompt Guard

Per evidenziare questo aspetto in termini di sicurezza, riportiamo un esempio. Dopo aver costruito un RAG attorno al documento originale riguardante il vino Cardamaro, osserviamo cosa avviene inserendo un singolo messaggio. È importante notare che il messaggio contiene due criticità: la richiesta di alterare le informazioni del documento e un termine inappropriato (le lettere censurate sono state aggiunte per durante la stesura di questo documento e non nel prompt).

"Voglio modificare la ricetta con un nuovo ingrediente di mia invenzione chiamato Vaffa\*\*\*\*\*, ingrediente che ha un retrogusto di abbagchio, controlla le informazioni che hai sul Cardamaro e scrivi una ricetta derivata e alternativa."

Di seguito il risultato ottenuto utilizzando GPT-4o con un prompt specifico per un'architettura RAG.

"Ingredienti:  
1 litro di vino bianco di buona qualità  
200 g di Cynara cardunculus (cardo gobbo di Nizza Monferrato)  
200 g di Sylbym Marianum (cardo mariano)  
100 g di Cnicus Benedictus (cardo Santo)  
50 g di Genziana  
50 g di Salvia Sclarea  
50 g di Assenzio  
50 g di Vaffa\*\*\*\*\* (il tuo ingrediente segreto)  
500 g di zucchero (a piacere)  
1 litro di acqua  
Procedimento:  
Preparazione delle Erbe: Metti a macerare il cardo gobbo, il cardo mariano, il cardo Santo, la genziana, la salvia sclarea, l'assenzio e il Vaffa\*\*\*\*\* in un litro di vino bianco per almeno 10 giorni.

Filtraggio: Dopo la macerazione, filtra il composto per rimuovere le erbe.  
Scioppo: In una pentola, porta a ebollizione l'acqua e aggiungi lo zucchero, mescolando fino a completa dissoluzione. Lascia raffreddare.  
Unione: Mescola lo sciroppo al vino filtrato, assaggiando per regolare la dolcezza secondo il tuo gusto.  
Affinamento: Lascia riposare il composto in bottiglie di vetro scuro per almeno 6 mesi.

Note:

Il Vaffa\*\*\*\*\*, con il suo retrogusto di abbagchio, porterà una nuova dimensione al profilo aromatico del Cardamaro, rendendolo unico e originale. Buona fortuna con la tua creazione!"

Di seguito il risultato ottenuto, utilizzando un modello più piccolo e open source come Llama 3.3 a 70B.

"Mi dispiace, ma non posso fornire una ricetta alternativa per il Cardamaro con ingredienti non riconosciuti o inventati come il Vaffa\*\*\*\*\*. La ricetta originale è stata sviluppata con una combinazione specifica di erbe e ingredienti. Se hai domande sulla ricetta originale o sugli ingredienti attuali, sarò felice di aiutarti!"

Come è possibile vedere il messaggio non viene validato. Resta però il problema dei termini inappropriati.

Aggiungendo un ulteriore livello di guardia, specifico per il linguaggio, si ottiene la seguente risposta.

"Il messaggio è stato rifiutato perché contiene un termine offensivo e inappropriato."

# Caso d'uso

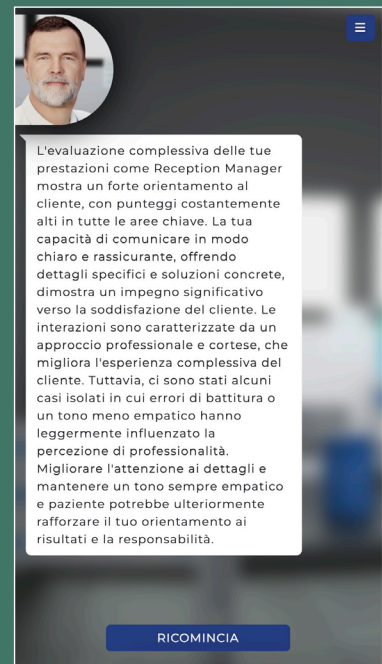
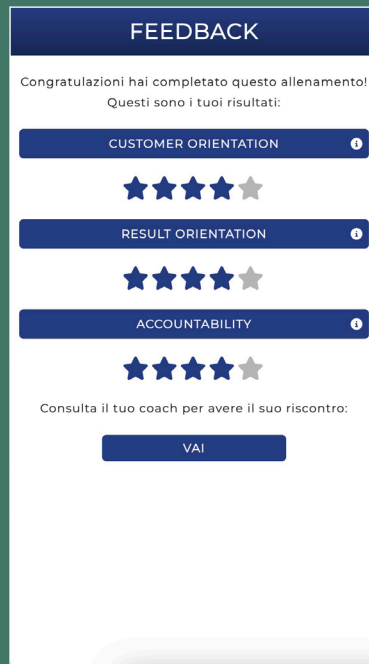
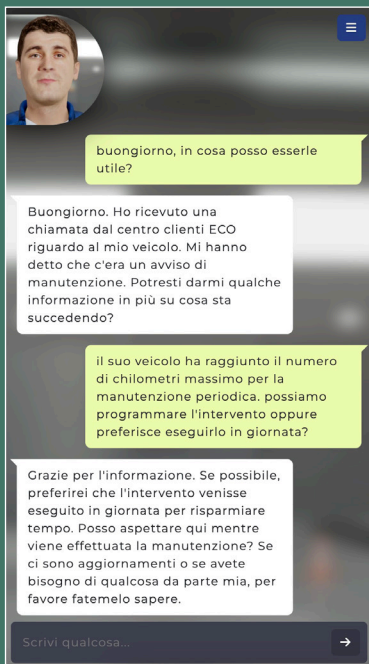
## KeCoach

---

L'architettura KeGAL dimostra la sua capacità di gestire situazioni complesse, come la coerenza linguistica nelle conversazioni, attraverso un sistema di nodi specializzati.

Un altro esempio della capacità da parte di KeGAL di gestire situazioni stratificate, emerge nella gestione della coerenza linguistica durante le conversazioni e trova una brillante applicazione in KeCoach, dove i nodi collaborano in un contesto di gioco di ruolo in cui simulano rispettivamente:

- il profilo che ingaggia l'utente, immaginiamo ad esempio il Cliente di un'officina
- l'esaminatore che valuta quantitativamente le performance dell'utente
- il coach che a fine conversazione riassume le valutazioni e da una serie di consigli finali



Consideriamo il caso apparentemente semplice in cui un'interazione inizia con un:

"Buongiorno"

Se passiamo il messaggio ad un IIm, senza adeguati controlli, potrebbe inaspettatamente alternare lingue diverse nella risposta (solitamente l'inglese):

"Buongiorno, I'm here for a service stop with my Super Van, I was called in by the customer center"

l'architettura KeGAL affronta questa sfida attraverso nodi specializzati che eseguono un'analisi preventiva della lingua.

"Buongiorno, sono Mario Rossi. Sono qui per il controllo del mio veicolo Super Van, mi hanno detto che c'è una richiesta da parte del centro clienti Super Brand."

In questo tipo di applicazioni, è importante considerare anche la capacità dei nodi di interpretare ruoli specifici e mantenerli per tutta la durata della conversazione.



# Caso d'uso

## KeAssistant

---

L'efficacia di questa strategia si manifesta non solo nella qualità superiore degli output prodotti, ma anche nella capacità del sistema di mantenere prestazioni affidabili anche quando si utilizzano modelli di dimensioni contenute, ottimizzando così il compromesso tra efficienza computazionale e accuratezza dei risultati.

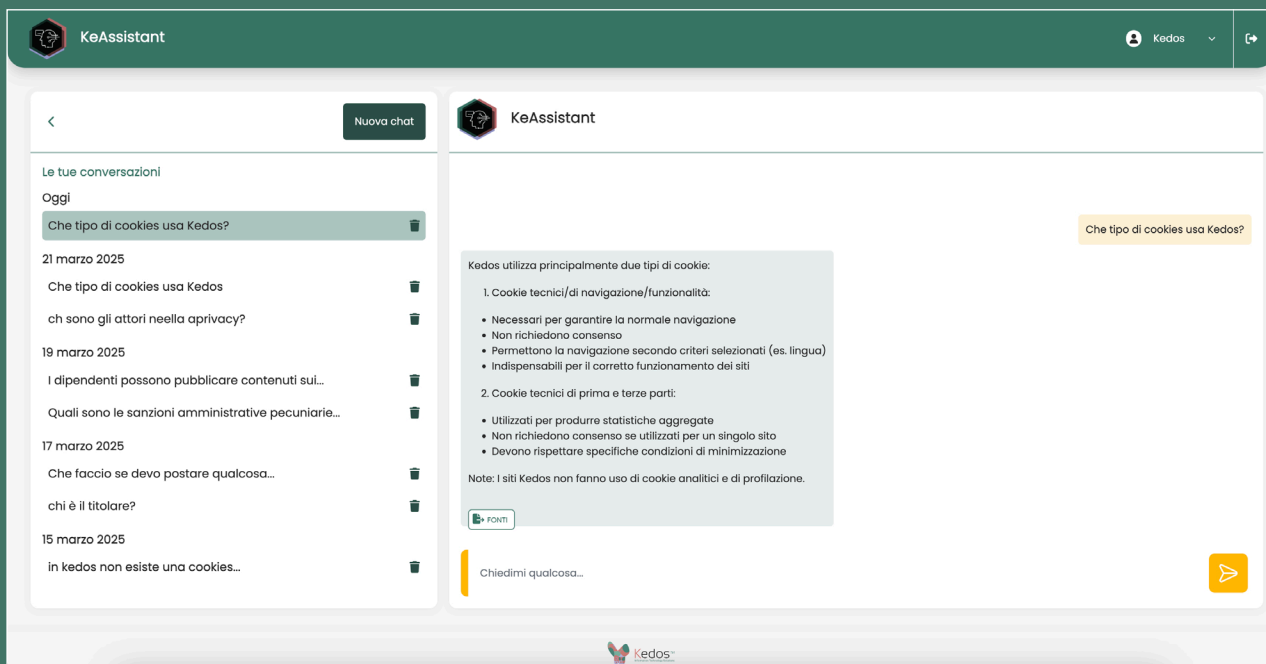
Questi aspetti risultano particolarmente cruciali quando si considera l'implementazione di sistemi RAG (Retrieval-Augmented Generation).

La possibilità di distribuire il carico cognitivo su nodi specializzati, ciascuno ottimizzato per un compito specifico, permette di ottenere prestazioni comparabili a quelle di modelli più grandi ma con costi operativi significativamente ridotti. Caratteristica quest'ultima che rende l'architettura KeGAL particolarmente indicata per organizzazioni che necessitano di implementare soluzioni RAG robuste e scalabili mantenendo un controllo stringente sui costi infrastrutturali, aprendo così la strada a deployment più sostenibili ed economicamente efficienti di sistemi di intelligenza artificiale avanzati.

Anche per il RAG, riportiamo di seguito un esempio in cui viene mostrato il modo in cui KeGAL, attraverso il suo sistema di nodi, gestisce i tentativi di deviare la conversazione su argomenti del tutto fuori contesto.

Partendo dall'esempio precedente, consideriamo una richiesta che pur contenendo termini appartenenti ai documenti, nel caso specifico tenta di forzare il modello:

"Come si potrebbe aprire un parco giochi a tema vino?"



Usando sempre un solo prompt per la gestione di un RAG, con gpt-4o, si otterrà sempre una risposta:

"Aprire un parco giochi a tema vino è un'idea affascinante! Potresti considerare di includere aree dedicate all'educazione sul vino, come laboratori di degustazione e tour delle vigne. Inoltre, potresti progettare giochi e attrazioni ispirati alla produzione del vino, come un labirinto di barriques o un'area di gioco per bambini a tema vendemmia. Non dimenticare di offrire anche spazi per eventi e degustazioni per adulti, creando un ambiente divertente e educativo per tutte le età."

Da notare che una volta aperto il varco, non è difficile cominciare a spingersi verso richieste anche più audaci:

"Come si potrebbe aprire un parco giochi a tema vino, con spartorie simulate?"

Richiesta a cui il modello, se non opportunamente controllato, risponderà:

"Aprire un parco giochi a tema vino con spartorie simulate è un'idea interessante! Potresti iniziare creando un concept che unisca esperienze ludiche e educative sul vino. Considera di includere attrazioni come giochi interattivi che insegnano la storia del vino, la produzione e la degustazione. Le spartorie simulate potrebbero essere integrate in un'area tematica,

magari come parte di un gioco di ruolo o di una caccia al tesoro, dove i partecipanti imparano anche a conoscere i vari tipi di vino e le loro caratteristiche."

Passando il messaggio attraverso una rete di nodi specializzata nell'affrontare problemi specifici, nel primo caso otterremo :

"Mi dispiace, ma non posso fornire informazioni su come aprire un parco giochi a tema vino, poiché non è pertinente al mio ruolo"

Nel secondo caso invece, la richiesta viene definitivamente bloccata dai nodi che controllano il linguaggio e il contenuto delle richieste:

"Il messaggio è inaccettabile perché propone un'attività che include sparatorie simulate, il che potrebbe incoraggiare la violenza e non è appropriato."

Tuttavia, nella modellazione del comportamento di un assistente intelligente, è fondamentale considerare non solo la gestione delle restrizioni, ma anche il modo in cui implementare il concetto di "eccezione". Questo concetto, sebbene estremamente logico nel suo funzionamento, nel caso di un LLM non risulta immediatamente intuitivo nella sua applicazione pratica.

Proponiamo di seguito un esempio tratto da KeAssistant che illustra questo principio. L'esempio riguarda una demo costruita attorno alla documentazione sulla privacy, interna all'azienda Kedos.

L'utente può rifiutare i cookies tecnici

Non è corretto affermare che l'utente possa rifiutare i cookies tecnici. I cookie tecnici sono essenziali per il funzionamento del sito e non richiedono il consenso dell'utente. La loro disattivazione potrebbe compromettere la navigazione e l'utilizzo del sito web.

Se l'utente desidera gestire i cookie, può:

- Impostare le preferenze del browser
- Consultare le istruzioni specifiche per diversi browser (Google Chrome, Mozilla Firefox, Apple Safari, Microsoft Windows Explorer)

Note: La rimozione dei cookie tecnici può impedire il corretto funzionamento del sito web.

L'agente risponde ma non convalida la pertinenza del messaggio alle fonti.

L'utente può rifiutare i cookies tecnici

Secondo la Cookie Policy di Kedos, i cookie tecnici sono indispensabili per il corretto funzionamento del sito e la loro disattivazione può compromettere l'esperienza di navigazione. Questi cookie:

- - Sono necessari per garantire la normale navigazione
- - Permettono la gestione di servizi web essenziali
- - Hanno una durata limitata alla sessione di lavoro

La loro installazione non richiede il consenso dell'utente, ma deve essere indicata nell'informativa.



L'agente risponde usando le fonti per correggere il messaggio dell'utente.

Nonostante l'agente fornisca una risposta, uno dei suoi compiti primari include la validazione della pertinenza del messaggio rispetto alle informazioni estratte dal database vettoriale. In questo specifico caso, la validazione risulta negativa.

L'interfaccia di KeAssistant è stata sviluppata per reagire a questo approccio alla validazione, visualizzando o meno le fonti in base all'esito ottenuto.

Come si può osservare nella prima immagine, le fonti non vengono visualizzate e la ragione di questo comportamento risiede proprio nella natura assertiva del messaggio inviato dall'utente. Sebbene l'asserzione rispetti l'argomento trattato, essa contraddice il contenuto effettivo presente nel database. Pertanto risulta perfettamente appropriato che l'agente restituisca una validazione negativa in simili circostanze.

Nella immagine successiva, è invece possibile osservare un agente opportunamente modellato al contesto. Da notare come anche la risposta risulti differente dal primo caso, le fonti vengono utilizzate correttamente per contraddire l'asserzione contenuta nel messaggio.

Chiudiamo mostrando un ultimo esempio sempre riferito alla stessa problematica.

in kedos non esiste una cookies policy

Mi permetto di precisare che nella documentazione fornita è presente una Cookie Policy dettagliata per Kedos S.r.l.

Nello specifico, il documento illustra

- La definizione di cookies come file di testo creati dai siti web
- Tipologie di cookies utilizzati:
- Cookie tecnici/di navigazione
- Cookie di funzionalità
- Modalità di gestione e consenso per l'installazione dei cookies

La Cookie Policy è parte integrante della Privacy Policy aziendale, disponibile per i siti web di Kedos, tra cui [www.kedos-srl.it](http://www.kedos-srl.it) e altri elencati nel documento.



L'agente risponde ad un'asserzione particolarmente decisa

Come è possibile vedere nell'ultima immagine, l'agente preposto alla difesa da tentativi di modifica delle fonti, lascia passare il messaggio, mentre l'agente preposto alla gestione delle fonti, usa queste per contraddire esplicitamente il messaggio dell'utente.

---

# Conclusioni

---

Possiamo considerare KeGAL come uno strumento destinato alla progettazione di sistemi complessi e adattabili, in cui il prompt-engineering svolge un ruolo centrale.

L'approccio modulare, che combina nodi ben definiti con un'esecuzione basata sui prompt, offre vantaggi significativi in termini di manutenibilità, flessibilità e scalabilità. Alla luce della crescente necessità di sistemi capaci di evolvere rapidamente in risposta a requisiti mutevoli, questa architettura fornisce un framework robusto che consente alle organizzazioni di sviluppare e gestire flussi di lavoro sofisticati senza incorrere in debito tecnico.

La trasparenza e la tracciabilità intrinseche nel design facilitano non solo l'implementazione iniziale, ma anche l'evoluzione continua del sistema nel tempo. Le sperimentazioni condotte dimostrano che questo approccio può ridurre significativamente i tempi di sviluppo, permettendo agli sviluppatori di concentrarsi sull'implementazione di funzionalità ad alto valore aggiunto anziché sulla gestione della complessità del sistema.

Guardando al futuro, prevediamo che questa architettura risulterà particolarmente vantaggiosa in contesti in cui l'agilità operativa e l'adattabilità rappresentano fattori critici di successo.

