



# LEGAL RAG

---

# ...CHATBOT...

---

Fabio Accurso - M63001723

Vittoria Alberto - M63001750

Benito Cervone - M63001747





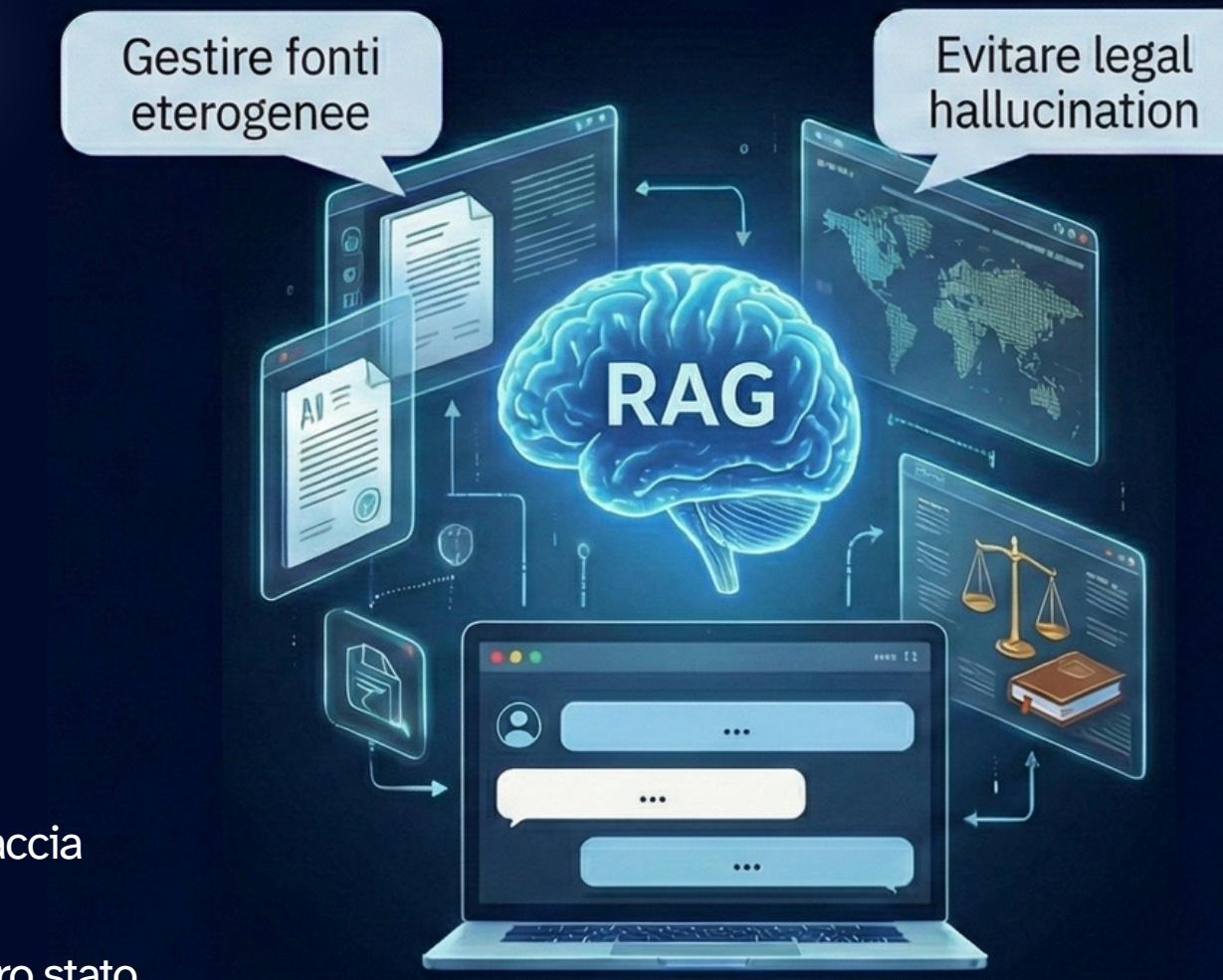
# PANORAMICA E OBIETTIVI DEL PROGETTO

## Obiettivo

Costruire un sistema RAG (Retrieval-Augmented Generation) capace di rispondere a quesiti su Divorzio e Successioni in tre giurisdizioni (Italia, Estonia, Slovenia).

## Sfide Chiave:

- Gestire fonti legali eterogenee (Codici Civili vs Casi Legali).
- Garantire che per ambiti legali appartenenti a quelli selezionati, inerenti alle tre nazioni di riferimento, faccia Retrieval e risponda sulla base dei documenti recuperati.
- Garantire che per una domanda su un determinato stato non vengano consultate fonti relative ad un altro stato (rischio di "legal hallucination").
- Garantire che il sistema risponda in maniera cordiale per argomenti non inerenti all'ambito legale.
- Il sistema si pronuncia in materia legale per paesi non appartenenti ai paesi selezionati senza però fare retrieval.
- Il sistema si pronuncia in materia legale per ambiti legali non appartenenti a quelli selezionati senza però fare retrieval.
- Confrontare due strategie di routing: Single-Agent (decisioni interne) vs Multi-Agent (orchestrazione gerarchica).



ITALIA



ESTONIA



SLOVENIA



# STRUTTURA DEL PROGETTO PYTHON

- `Home.py` Script principale del progetto; entry point che eseguiamo tramite Dockerfile
- `requirements.txt` Lista di pacchetti Python (e versioni) necessari per eseguire il progetto
- `.env` File contenente le chiavi private di HuggingFace e OpenAI
- `Contest_Data` Cartella dove sono conservati i file di input (JSON)
- `.gitignore` Dice a Git quali file/cartelle deve ignorare (.env, Contest\_Data)
- `build_vector_stores.py` Fa embedding e crea quattro database vettoriali FAISS separati
- `docker_compose.yaml` Indica a Docker di costruire l'immagine del container, esporre la porta 8501 e indica il nome del file dove sono presenti le chiavi private.



▼ TEXTMINING-MAIN

- ▼ backend
  - > \_\_pycache\_\_
  - ➊ \_\_init\_\_.py
  - ➋ config.py
  - ➌ document\_loader.py
  - ➍ embeddings.py
  - ➎ llm\_provider.py
  - ➏ rag\_multiagent.py
  - ➐ rag\_pipeline.py
  - ➑ rag\_single\_agent.py
  - ➒ rag\_utils.py
  - ➓ vector\_store.py
- > Contest\_Data
- ▼ pages
  - ➊ Chatbot.py
  - ➋ Evaluation.py
- > vector\_store
- ➊ .env
- ➋ .gitignore
- ➌ build\_vector\_stores.py
- ➍ docker-compose.yaml
- ➎ Dockerfile
- ➏ Home.py
- ⠁ requirements.txt

# RAG: SCHEMA GENERALE

- **Root project block (TEXTMINING-MAIN/)**

“Shell” dell’applicazione complessiva: come viene avviato, installato, containerizzato e configurato il sistema. È il wrapper operativo che racchiude tutto il resto.

- **Backend block (backend/)**

Il motore centrale del sistema: dove risiedono la configurazione, i modelli, il retrieval, la logica del database vettoriale e il ragionamento RAG/agente. Questo blocco decide come vengono prodotte le risposte.

- **Pages block (pages/)**

Il livello di interazione: pagine Streamlit che consentono a un utente di configurare il sistema, creare indici, chattare con l’agente e monitorare il comportamento. Espone il backend all’utente.

- **Vector store block (vector\_store/)**

La base di conoscenza indicizzata: database vettoriali persistenti (FAISS) creati dai corpora, pronti per una rapida ricerca di somiglianze durante il RAG.

- **Data block (Contest\_Data/)**

Fonte delle informazioni grezze: corpora JSON originali da cui vengono creati i vector store.

- **Setting block**

File di impostazione (.gitignore, .env, requirements, Dockerfile, docker-compose)



# DATASET DI RIFERIMENTO

Contest\_Data > Italy > Divorce\_italy > Article\_164.json > ...

```
1  {  
2    "content": "\nARTICLE 164\nSIMULATION OF MARRIAGE AGREEMENTS\nEvidence of the simulation of marriage  
3    "metadata": {  
4      "civil_codes_used": "Art. 164",  
5      "law": "Divorce",  
6      "type": "ITALY"  
7    }  
8 }
```

Contest\_Data > Italy > Italian\_cases\_json\_processed > 2. Case 88\_2016.json > ...

```
1 {  
2   "content": "this legal text provides an overview of the progress of a legal case without mentioning the names of  
3   "metadata": {  
4     "CASE_ID": "88/2016",  
5     "civil_codes_used": [  
6       "Art. 1111"  
7     ],  
8     "cost": "1'167,13 euro compensation",  
9     "duration": "60 days",  
10    "law": "Inheritance",  
11    "state": "ITALY",  
12    "type": "extrajudicial",  
13    "succession_type": "testamentary",  
14    "subject_of_succession": "properties",  
15    "disputed_issues": "division of assets",  
16    "relationship_between_parties": "co-owners",  
17    "number_of_persons_involved": 10  
18  }  
19 }
```

## Struttura dei dati

I dati di input si trovano nella cartella Contest\_Data e sono strutturati in tre sottocartelle (una per ogni stato), ciascuna delle quali si divide in altre tre sottocartelle (articoli divorzio, articoli successione e casi passati).

```
└─ Contest_Data  
    └─ Estonia  
        ├─ Divorce_estonia  
        ├─ Estonian_cases_json_processed  
        └─ Inheritance_estonia  
    └─ Italy  
        ├─ Divorce_italy  
        ├─ Inheritance_italy  
        └─ Italian_cases_json_processed  
    └─ slovenia  
        ├─ Divorce_slovenia  
        ├─ Inheritance_slovenia  
        └─ Slovenian_cases_json_processed
```

## Formato dei dati

I dati sono in formato JSON. Questo è cruciale perché, come vedremo più avanti, il nostro sistema non legge solo il testo, ma sfrutta pesantemente i metadati (come il tag 'Country' o 'Law Area') per filtrare le informazioni ed evitare, ad esempio, di rispondere a una domanda sulla legge italiana usando una sentenza estone.



# DATA PROCESSING & GRANULAR VECTOR STORES

## Strategia di Indicizzazione (build\_vector\_stores.py)

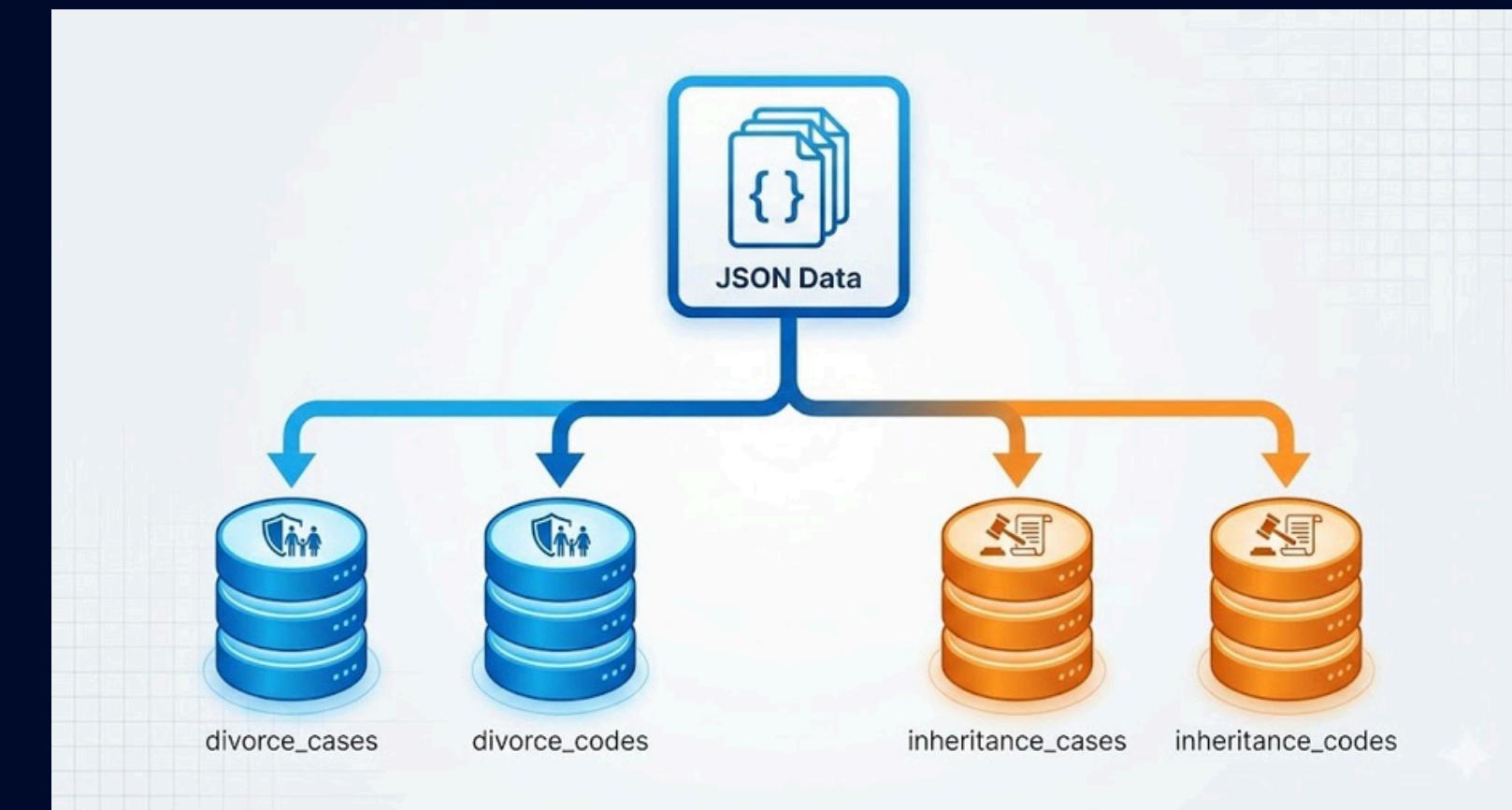
- Abbandono dell'approccio “indice unico” in favore di una segmentazione granulare.
- Creazione di 4 Vector Stores distinti e specializzati:
  1. divorce\_codes (Normative sul divorzio)
  2. divorce\_cased (Casi di divorzio)
  3. inheritance\_codes (Normative sulle successioni)
  4. inheritance\_cases (Casi di successioni)

## Implementazione Tecnica:

- Embedding Model: all-MiniLM-L6-v2 (Open Source)
- Caricamento: Separazione automatica basata sui metadati doc\_type (“code” vs “case”)

## Vantaggio Architetturale:

- Riduzione del “rumore semantico”: impedisce al sistema di confondere un articolo di legge con una sentenza specifica durante il retrieval.
- Abilita un routing preciso per gli agenti





# MODELLI E STACK TECNICO

## Embedding Model

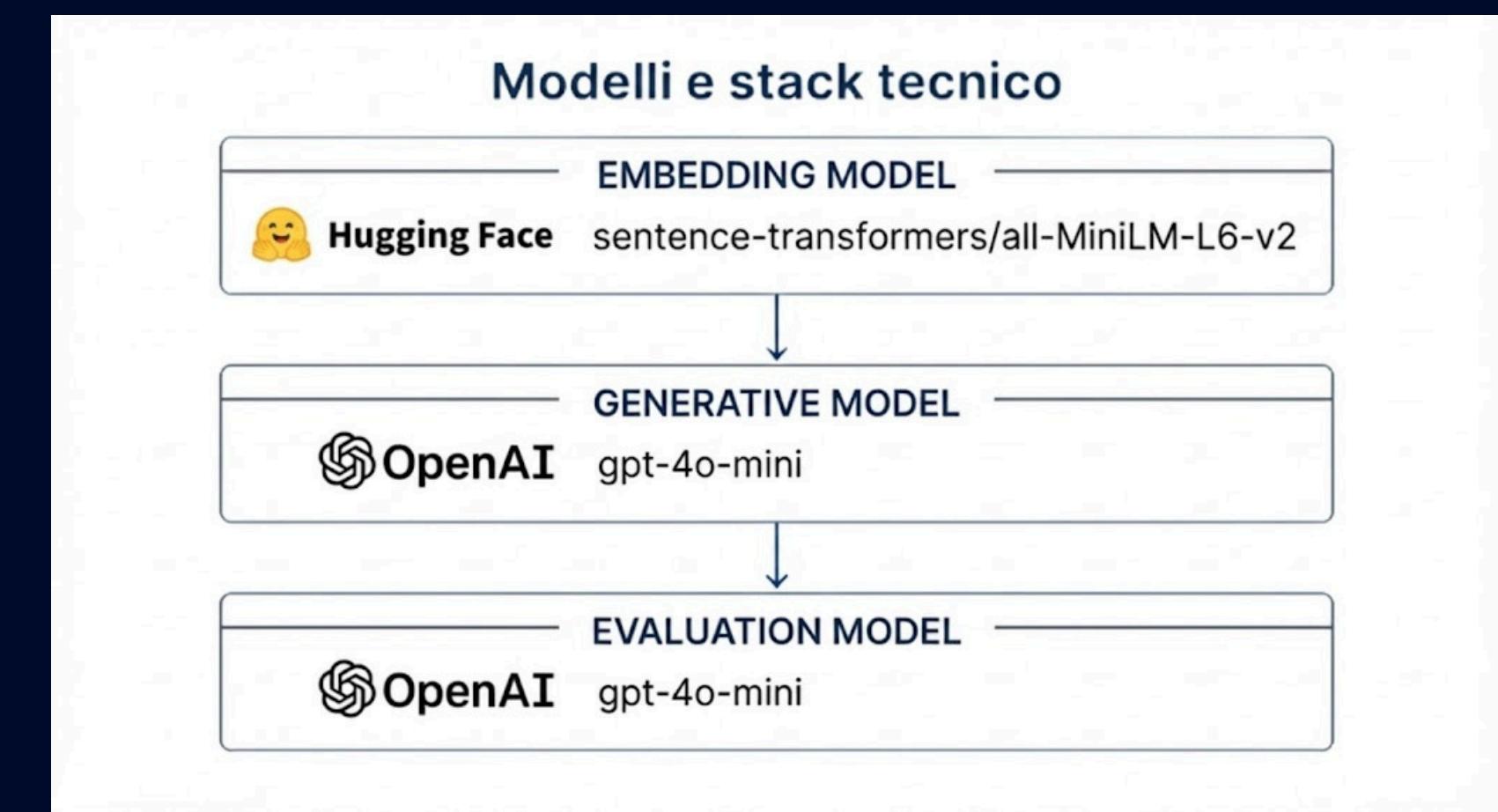
- Modello: sentence-transformers/all-MiniLM-L6-v2 (Open Source)
- Motivazione: La scelta è ricaduta su questo modello per il suo eccellente bilanciamento tra velocità di inferenza e capacità semantica. Essendo un modello estremamente leggero, è CPU-friendly e permette la creazione rapida degli indici vettoriali senza richiedere hardware dedicato costoso.

## Generative Model

- Modello: gpt-4o-mini
- Parametri: Temperatura impostata a 0.2
- Motivazione: Gpt-4o-mini è stato scelto come compromesso ideale tra costi contenuti, velocità di esecuzione e accuratezza analitica, spesso comparabile a modelli maggiori in task di valutazione strutturata. L'utilizzo di una temperatura bassa (0.2) è stato deliberato per minimizzare le "allucinazioni", forzando il modello a aderire strettamente al contesto legale recuperato dai documenti.

## Evaluation Model

- Modello: gpt-4o-mini
- Parametri: Temperatura impostata a 0
- Motivazione: L'utilizzo del pattern LLM-as-a-Judge richiede un modello con forti capacità di ragionamento per valutare metriche complesse come Faithfulness e Context Precision. La temperatura a zero garantisce il determinismo e la riproducibilità dei punteggi di valutazione.



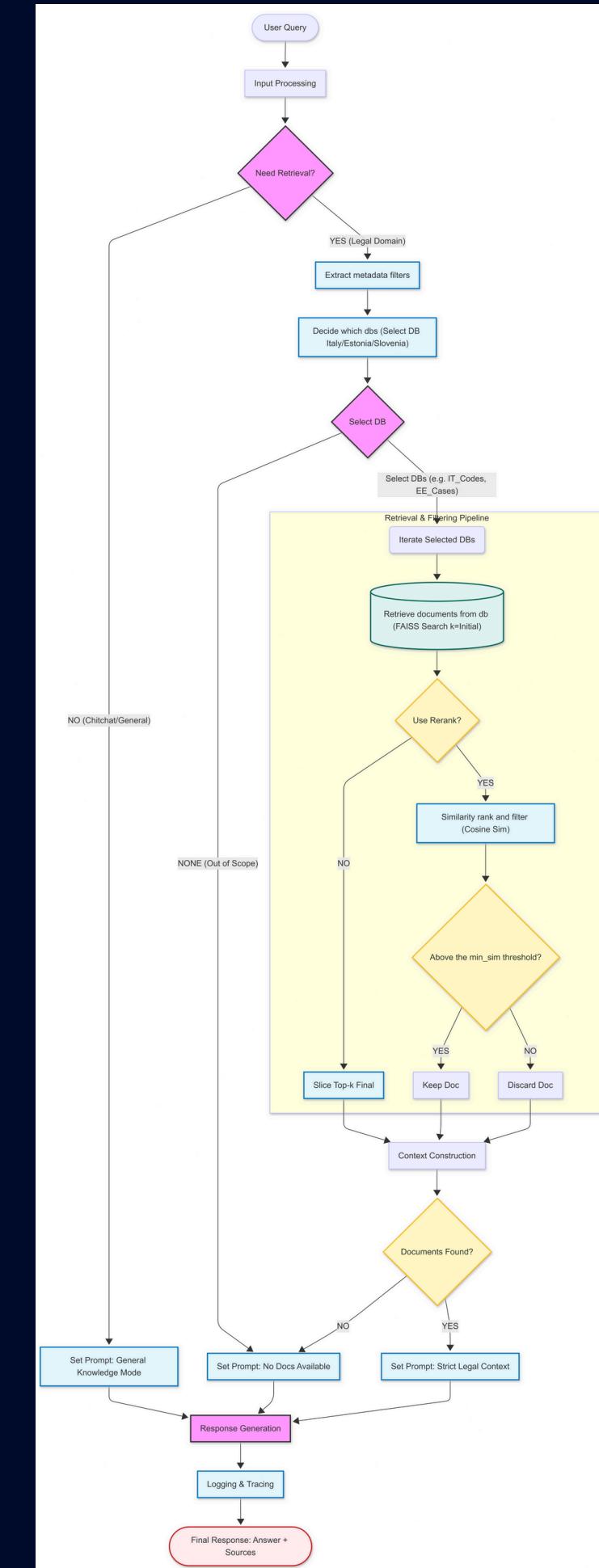


# TASK A: SINGLE-AGENT (REACT)

L'agente in accordo con il paradigma ReAct (Reasoning + Acting) non risponde subito, ma segue un ciclo di ragionamento.

- Thought: Analisi della domanda e decisione riguardo se attivare il retrieval.
  - Action: Selezione dinamica dei DB e applicazione filtri (es. Country=Italy).
  - Observation: Recupero dei documenti e lettura del contesto.
  - Answer: Generazione della risposta finale citando le fonti.

Prima dell'interrogazione ai database, la richiesta viene elaborata in due fasi: inizialmente un Router LLM distingue l'intento dell'utente (separando Chitchat da quesiti Legali). Successivamente, per le sole domande legali, vengono attivate le funzioni di utilità in `rag_utils.py` per l'estrazione dei filtri metadati, isolando le giurisdizioni supportate (Italia, Estonia, Slovenia) e le materie di competenza (Divorzio, Successioni).





# TASK B: MULTI-AGENT (SUPERVISOR)

A differenza del Single-Agent, che fa tutto da solo, nel Multi-Agent c'è una gerarchia.

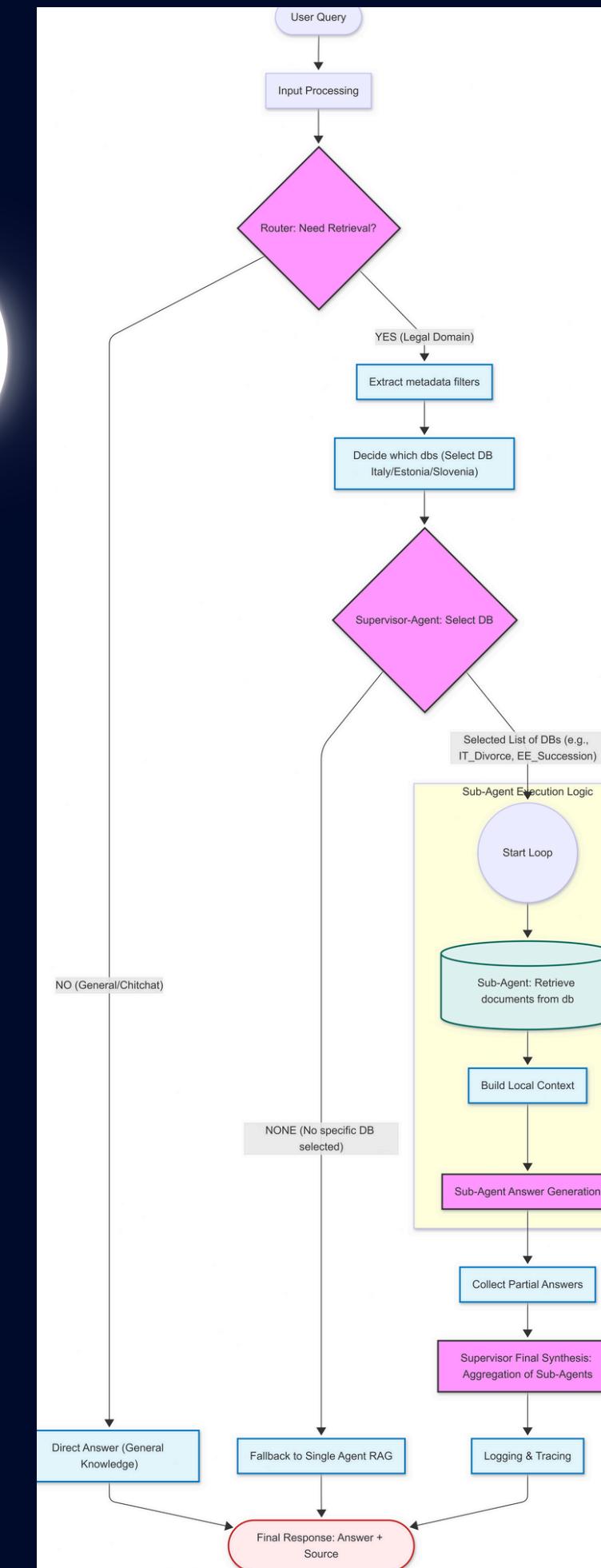
C'è un Supervisore che agisce come un manager. Quando arriva una domanda, il Supervisore non cerca subito i documenti.

Consulta le 'descrizioni' dei 4 database specializzati (introdotti in precedenza) e decide chi è l'esperto più adatto a rispondere.

In questo contesto se la domanda è complessa, ad esempio un confronto tra Italia ed Estonia, il Supervisore attiva entrambi i sotto-agenti, raccoglie le loro risposte parziali e le sintetizza. Questo approccio è molto più scalabile: il Supervisore non deve conoscere ogni singola legge, deve solo sapere chi interrogare.

Il flusso seguito è dunque il seguente:

- Delegation: Il Supervisore invia la query ai sotto-agenti selezionati.
- Execution: Ogni sotto-agente esegue il retrieval sul proprio indice specifico.
- Synthesis: Il Supervisore raccoglie le risposte parziali e le unifica in una risposta finale coerente, risolvendo eventuali conflitti.





Home Chatbot Evaluation Deploy :

## Legal RAG System

Multi-Agent Retrieval-Augmented Generation for Cross-Border Legal Knowledge

### Benvenuto

Questo sistema utilizza Retrieval-Augmented Generation (RAG) e architetture agentiche per rispondere a domande legali complesse su:

- Italia - Divorzio e Successioni
- Estonia - Divorzio e Successioni
- Slovenia - Divorzio e Successioni

Il sistema combina tecniche avanzate di recupero di informazioni con modelli di linguaggio per fornire risposte accurate e ben documentate su normative e giurisprudenza in materia di diritto di famiglia e successioni.

### Navigazione

- Chatbot Interface
- Evaluation Dashboard

Caratteristiche:

- Configurazione parametri di retrieval
- Visualizzazione delle fonti citate
- Tracciamento del reasoning (opzionale)
- Export della conversazione in JSON

Vai alla pagina Chatbot nella sidebar

Vai alla pagina Evaluation nella sidebar

# INTERFACCIA HOME... ...E CHATBOT

Home Chatbot Evaluation Deploy :

## Legal RAG Chatbot

Ask questions about divorce and inheritance law across Italy, Estonia, and Slovenia

Tip: Toggle 'Show reasoning trace' in the sidebar to see how the agent makes decisions.

### Configuration

Select agent type:  Single Agent (ReAct)  Multi-Agent (Supervisor)  Show reasoning trace

### Retrieval Parameters

Initial retrieval (top\_k): 20

Final documents (top\_k\_final): 5

Enable similarity reranking

### Model Info

LLM: gpt-4o-mini  
Embeddings: sentence-transformers/all-MiniLM-L6-v2  
Vector Store Info

Ask a legal question...



# VALUTAZIONE INTERNA

## Risultati del RAG single-agent (ReAct)

Risultati Valutazione Aggregati					
Faithfulness	Answer Relevancy	Context Precision	Context Recall	Answer Correctness	
0.69	0.35	0.64	0.77	0.37	
Tabella Dettagliata per Interazione					
Query ID	faithfulness	answer_relevancy	context_precision	context_recall	answer_correctness
1	0	0.8764	0	0	0.5664
2	0.9615	0.855	1	1	0.3647
3	1	0	1	0.8571	0.4845
4	0.8	0	1	1	0.2379
5	0.6667	0	0.2235	1	0.2021

## Risultati del RAG multi-agent (Supervisore)

Risultati Valutazione Aggregati					
Faithfulness	Answer Relevancy	Context Precision	Context Recall	Answer Correctness	
0.56	0.86	0.63	0.80	0.70	
Tabella Dettagliata per Interazione					
Query ID	faithfulness	answer_relevancy	context_precision	context_recall	answer_correctness
1	0	0.8151	0	0	0.9504
2	1	0.8117	1	1	0.6312
3	0.9091	0.9721	1	1	0.5548
4	0	0.701	1	1	0.6079
5	0.8667	0.9887	0.1393	1	0.7459

- Answer Relevancy e Correctness:** Il sistema Multi-Agente dimostra prestazioni nettamente superiori rispetto all'Agente Singolo. Ciò suggerisce che, sebbene l'agente singolo fosse in grado di recuperare informazioni, il ciclo ReAct ha faticato a sintetizzare una risposta coerente di fronte a scenari legali sfaccettati o confronti tra paesi.
- Prestazioni di Retrieval (Context Recall & Precision):** Entrambi i sistemi hanno mantenuto standard di recupero comparabili. Questo indica che il problema principale dell'Agente Singolo non risiedeva nell'incapacità di trovare i documenti, ma nella minor capacità di utilizzarli efficacemente nella fase di generazione.
- Faithfulness:** L'Agente Singolo ha ottenuto un punteggio di fedeltà leggermente superiore rispetto al Multi-Agente. Questa discrepanza può essere attribuita al passaggio di aggregazione del Supervisore, che sintetizza le risposte parziali degli agenti specializzati. Tale processo potrebbe astrarre leggermente dal testo sorgente diretto rispetto a una generazione immediata, pur risultando in una risposta finale molto più corretta e pertinente.
- Scalabilità e Robustezza:** Il sistema Multi-Agente garantisce migliore scalabilità (aggiunta di un nuovo paese) e, allo stesso tempo, migliore robustezza (non confusione delle query).



# VALUTAZIONE UFFICIALE

## Aggregated (mean) scores ↗

context\_precision: nan

context\_recall: 0.952

faithfulness: 0.808

answer\_relevancy: 0.825

answer\_correctness: 0.613

- Context Precision (NaN): Il valore "not a number" è un indicatore positivo in questo contesto specifico. Poiché 2 domande su 10 sono state ritenute fuori dominio e il sistema ha correttamente deciso di non effettuare il retrieval (routing verso la conoscenza interna o rifiuto), il calcolo matematico della precisione su base documentale non è applicabile per quei turni. Questo conferma l'efficacia della logica ReAct/Supervisor nel filtrare richieste non pertinenti.
- Context Recall (0.952): È la metrica più alta. Dimostra che il sistema è estremamente efficace nel reperire le informazioni necessarie all'interno del dataset multinazionale per rispondere correttamente ai quesiti legali.
- Faithfulness (0.808): Un punteggio solido che garantisce la trasparenza del sistema. Indica che le risposte generate sono strettamente ancorate ai documenti legali recuperati (Codici Civili e casi passati), riducendo drasticamente il rischio di allucinazioni.
- Answer Relevancy (0.825): Le risposte fornite sono altamente pertinenti alle intenzioni dell'utente, dimostrando che il modello riesce a processare correttamente la complessità delle domande legali riguardanti Italia, Estonia e Slovenia.
- Answer Correctness (0.613): Pur essendo un risultato positivo, rappresenta la sfida maggiore. Questa metrica confronta la risposta con la ground truth. Il punteggio suggerisce che, pur essendo i dati corretti e fedeli, la formulazione giuridica può presentare sfumature diverse rispetto alla soluzione ideale attesa, un aspetto tipico nel dominio del diritto civile.