DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

UNIVERSITÀ DEGLI STUDI DI PADOVA – DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
CORSO DI LAUREA IN INGEGNERIA INFORMATICA

RELAZIONE PER LA PROVA FINALE

AI CHATBOT FOR DEVELOPER SUPPORT

RELATORE: Prof. Emanuele Menegatti

CORRELATORE: Dott. Cristian Pirlog

LAUREANDO: Tommaso Dainese

MATRICOLA: 2102606

AI CHATBOT FOR DEVELOPER SUPPORT

Padova, 21/07/2026

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

PROJECT GOALS

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

OBIETTIVO

Creare un chatbot intelligente capace di interrogare diverse fonti documentali private, interne all'azienda, e restituire risposte utili, precise e basate sui contenuti realmente disponibili.

PRODUTTIVITA' MIGLIORATA

Fornire risposte immediate e accurate, riducendo i ritardi causati dalla ricerca nella documentazione.

ACCURATEZZA E AFFIDABILITÀ:

Garantire che tutte le risposte siano corrette, complete e coerenti, minimizzando il rischio di interpretazioni errate.

CENTRALIZZAZIONE DELLA CONOSCENZA:

Creare una fonte di conoscenza unificata rendendo le informazioni facilmente accessibili e collegate.

DESCRIZIONE

La struttura del progetto si divide in flussi distinti a seconda dell'esigenza dell'utente:

- Risposta basata su documentazione precedentemente estratta e elaborata localmente in un'unico spazio:

OFFLINE RAG

Retrieval **A**ugment **G**eneration:
tecnica di intelligenza artificiale che permette di recuperare dati da fonti esterne affidabili (documenti, database, web) prima di generare una risposta

✓
- Preciso
- Veloce
- Offline

✗
- Necessita aggiornamenti periodici della documentazione
- Occupa spazio locale

DESCRIZIONE

- Risposta basata su documentazione estratta in real-time da vari provider:

LIVE MCP

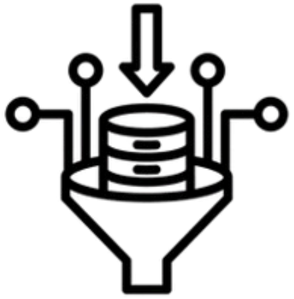
Model **C**ontext **P**rotocol:
programma che agisce da intermediario tra i modelli AI e fonti dati esterne, standardizzando il modo in cui condividono informazioni. Permette di accedere a file, database e API

- Collegato ai provider in cloud (no aggiornamenti richiesti)
- Non occupa spazio locale

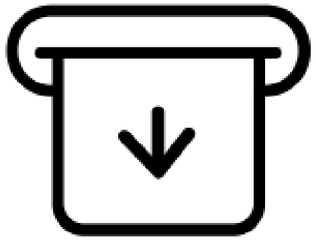
- Più lento
- introduce maggiori dipendenze

ARCHITETTURA

L'architettura è pensata in modo modulare, collegando, ma mantenendo indipendenti i seguenti stadi:



INJESTION



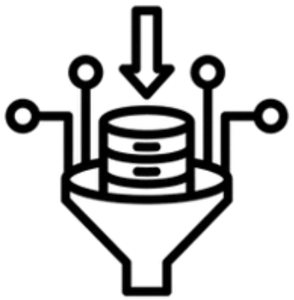
RETRIEVAL

GENERATION

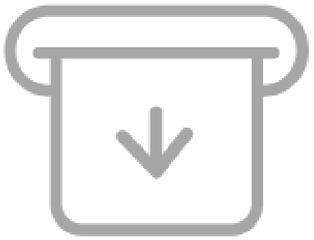
STORAGE

ARCHITETTURA

Fase di injestion



INJESTION



RETRIEVAL

GENERATION

STORAGE

INJESTION

Nella fase di injestion vengono reperiti e memorizzati localmente i documenti da diversi provider

1. **RACCOLTA**
   Il sistema raccoglie i documenti accessibili dalle sorgenti scelte.
2. **CLEANUP**
   Pulizia, normalizzazione e preprazione del testo
3. **CHUNKING**
   Il testo viene suddiviso in blocchi più piccoli e arricchito di metadati
4. **STORAGE**
   I chunk vengono salvati in locale in due maniere differenti

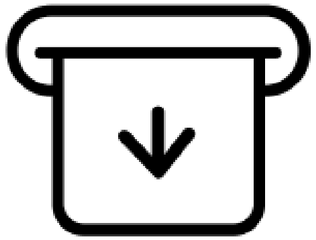
Vector Database:
I chunk vengono salvati sottoforma di embedding (sequenza di numeri)

Database tradizionale:
I chunk sono salvati come testo insieme ai loro metadati.

ARCHITETTURA

Fase di retrieval

INJESTION



RETRIEVAL

GENERATION

STORAGE

RETRIEVAL

1. **QUERY**

   Viene costruita la query (arricchita con i precedenti messaggi dell'utente per garantire coerenza tra i messaggi) con cui interrogare la documentazione

2. **RICERCA**

   La query e la documentazione vengono confrontate ricercando documenti rilevanti con due strategie:

   - **Dense search** → confronto di similarità sugli embedding - Buona precisione con rilevanze semantiche
   - **Sparse search** → confronto di similarità sui chunk - Buona precisione con rilevanze lessicali

RETRIEVAL

3. **MERGING**

I risultati ottenuti dai due approcci vengono fusi in un'unica lista garantendo al contempo rilevanza semantica e lessicale

4. **MMR**

I chunk passati dal merge vengono sottoposti a **MMR** → MMR è un algoritmo che permette di eliminare ridondanze e informazioni duplicate

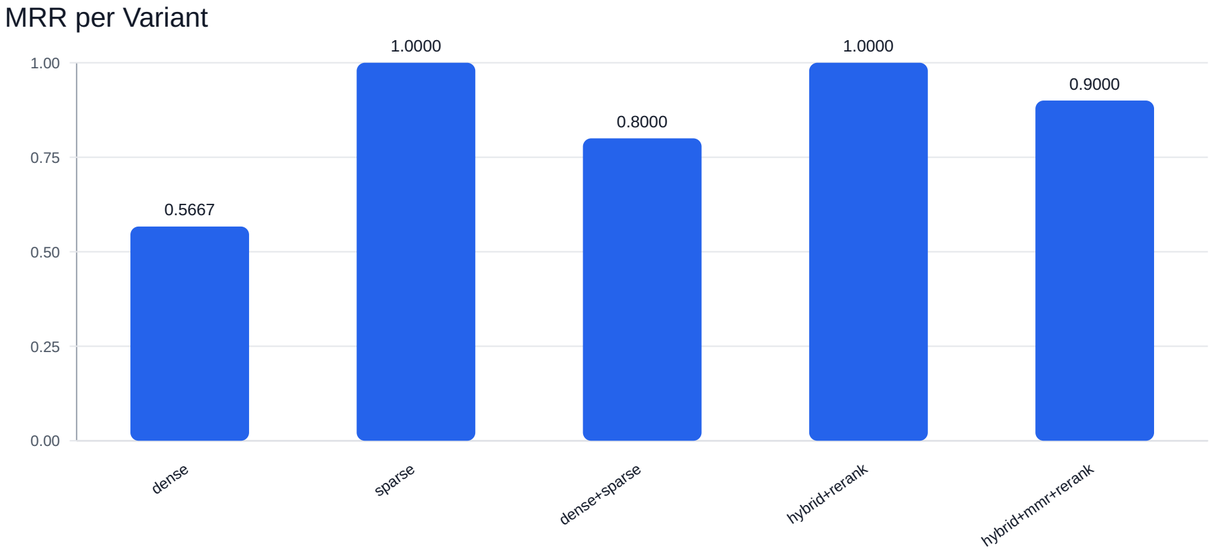
5. **RERANKING**

I chunk passano per un ultimo layer di confronto con un **reranker** e i più rilevanti finali vengono usati per costruire il contesto → Un reranker è un modello AI fine tuned per ordinare chunk rispetto alla loro rilevanza con una data query

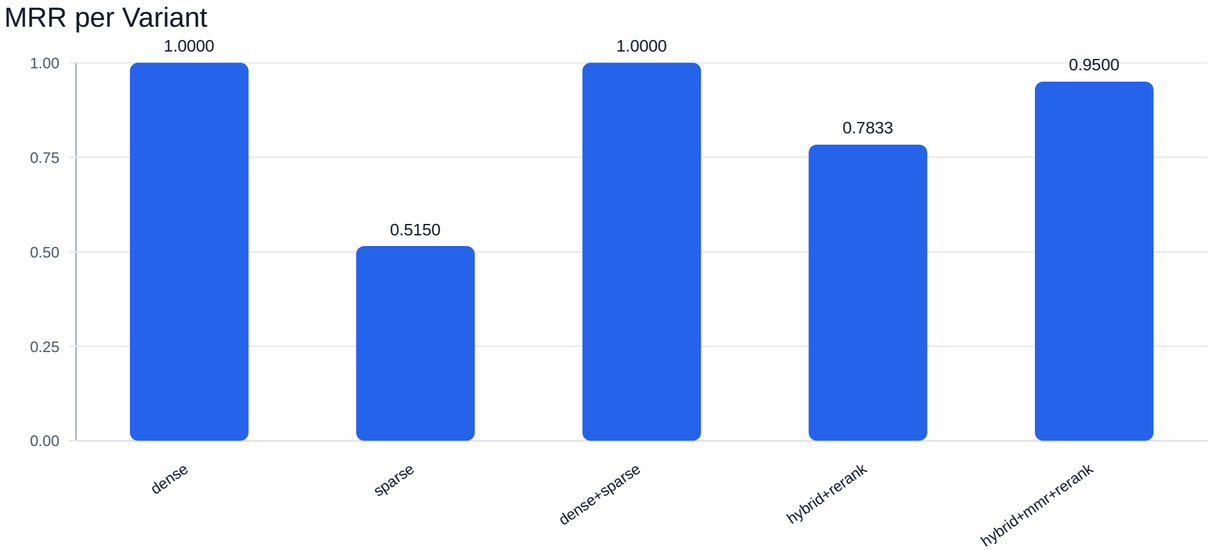
DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

BENCHMARKS

UNIVERSITÀ
DEGLI STUDI
DI PADOVA



SPARSE DOMINANT QUERIES

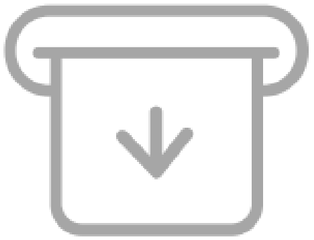


DENSE DOMINANT QUERIES

ARCHITETTURA

Fase di generation e storage

INJESTION



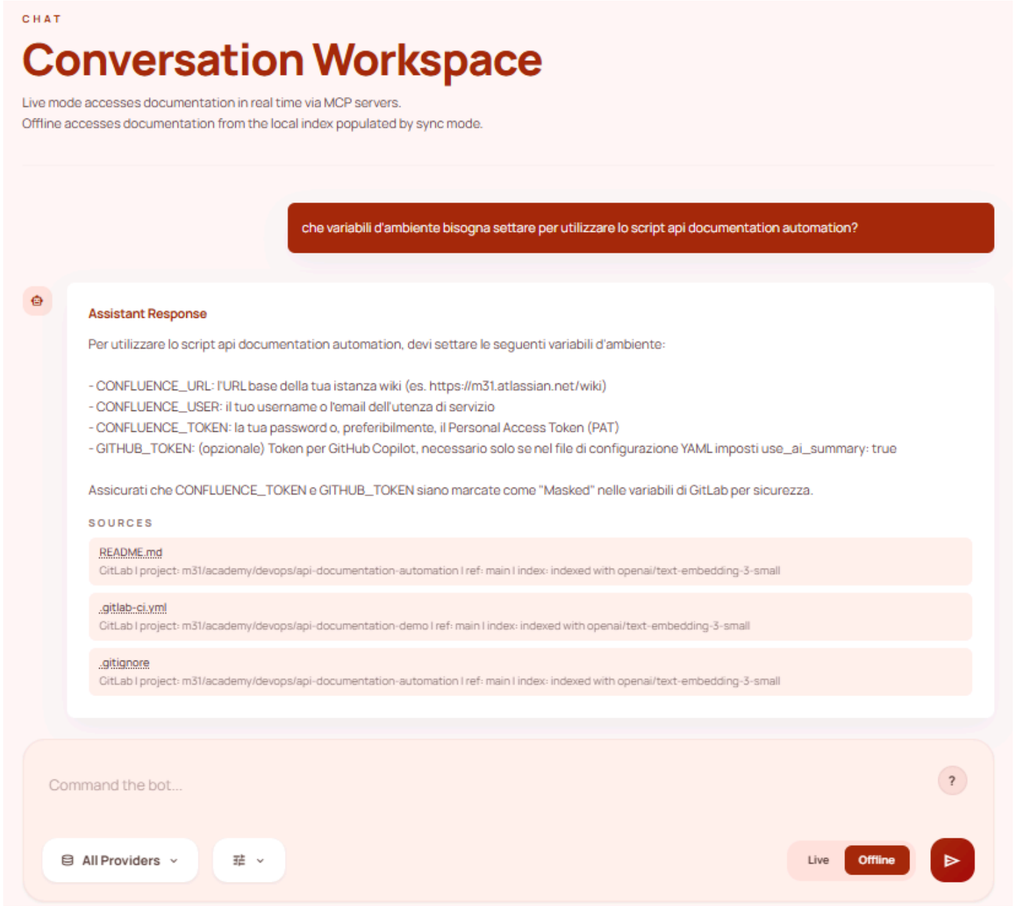
RETRIEVAL

GENERATION

STORAGE

GENERATION & STROAGE

1. **CONTESTO**
   Viene creato il contesto utilizzando i messaggi nella conversazione e i chunk estratti nel flusso di retrieval
2. **PROMPT**
   Il contesto viene arricchito di instruzioni per l'LLM dando origine al prompt finale
3. **GENERAZIONE**
   L'LLM riceve il prompt con query utente, contesto e istruzioni e risponde all'utente
4. **STORAGE**
   La conversazione e il messaggio vengono salvati per future riletture e interrogazioni



TOOLS

CODE



STORAGE


AI



IMPLEMENTAZIONI FUTURE

1. migrazione della UI a un framework per frontend, per renderlo più modulare e scalabile
2. introduzione di autenticazione e autorizzazione, così ogni utente vede solo i contenuti permessi
3. aggiunta di altri provider di documenti cloud
4. miglioramento dei flussi - gestione del contesto, recupero della cronologia conversazionale e routing piu intelligente delle query