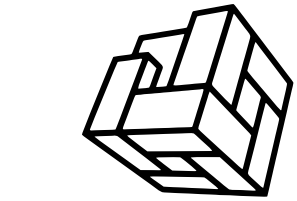
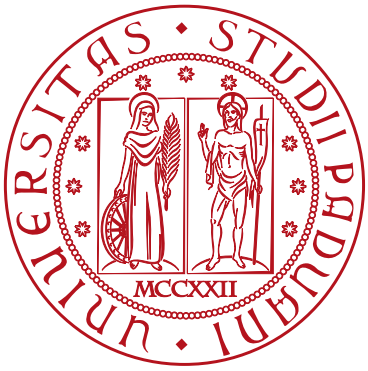


PEBKAC
Gruppo: 11
Email: pebkacswe@gmail.com
Docs: https://pebkac-swe-group-11.github.io
GitHub: https://github.com/PEBKAC-SWE-Group-11



Università degli Studi di Padova
Corso di Laurea: Informatica
Corso: Ingegneria del Software
Anno Accademico: 2024/2025

Deliverables per il Proponente

Informazioni sul documento:

Uso	Esterno
Destinatari	Vimar S.p.A.

Abstract:

Documento contenente i deliverables documentali richiesti dal Proponente

Indice

Elenco delle tabelle

1 Analisi

1.1 Risorse hardware

Il requisito che più ha complicato la realizzazione del prodotto è stata la necessità di un'applicazione che lavorasse in locale e quindi con risorse hardware limitate, soprattutto considerando che nelle fasi di realizzazione e test le risorse sarebbe state ancora più limitate, lavorando su dei portatili. Per superare queste difficoltà durante i test per la scelta del modello, si è optato per un LLM relativamente leggero e sebbene questa scelta abbia permesso di ridurre l'impatto sulle risorse hardware, ha anche comportato un possibile compromesso sulla precisione delle risposte generate. Si è inoltre riconosciuta la necessità di approfondire l'influenza del chunking sulle prestazioni del modello, che potrebbe avere un impatto negativo, soprattutto sui modelli meno performanti. Questo ci ha portato alla definizione di un **ambiente minimo di esecuzione del prodotto**:

CPU	Ryzen 7 7840HS
RAM	32GB DDR5
GPU	(Non utulizzata) Integrata, Radeon 780M Graphics

Tabella 1: Risorse Hardware Minime

Chiaramente con la possibilità di fornire maggiori risorse hardware si ha la possibilità di scegliere un LLM di dimensioni maggiori e quindi con capacità maggiori, portando idealmnente anche a un sensibile miglioramento.

1.2 Risposte del modello

Come descritto nella sezione precedente le limitate risorse hardware hanno influito sulla scelta del LLM: è stato scelto il migliore modello tra quelli in grado di funzionare con risorse e contesto limitati, tra i modelli valutati (varie versioni di Llama, Mistral, Qwen2.5, Deepseeck e Teuken - OpenGPT-X) alla fine si è optato per LLama3.1:8b. Gli stessi requisiti ci hanno anche spinto all'ottimizzazione di tutto il processo si indicizzazione dei dati, vettorizzazione e RAG. Il nostro algoritmo infatti cerca di fornire il contesto più utile possibile, poco vasto e molto pertinente, per aiutare il modello a generare una buona risposta. Al termine del progetto il prodotto funziona con le seguenti caratteristiche:

Modello	Llama3.1:8b
Chunk size	500
Overlap	50x2
Prodotti estratti	5
Chunk embeddati	3

Tabella 2: Modello LLM e RAG

Si possono trovare delle domande di prova con annessa valutazione nel Questionario di valutazione della qualità delle risposte compilato secondo le vostre indicazioni (Istruzioni del questionario di valutazione della qualità delle risposte) che mostrano come la qualità delle risposte generate dal prodotto finito sia stata oggetto di valutazione dettagliata, rivelando una **variabilità nella precisione, completezza e pertinenza delle informazioni** fornite dal modello LLM in diverse situazioni. Le valutazioni considerano la

corrispondenza con la risposta attesa, l'accuratezza dei dettagli e la chiarezza dell'output. Questo processo di valutazione è cruciale per identificare i punti di forza e le debolezze del sistema di interrogazione basato su LLM e per guidare i miglioramenti necessari.

1.3 Impatto green

La maggior parte della potenza di calcolo, e quindi del consumo, richiesta dall'app dipende dal modello di linguaggio usato (Llama3.1, versione da 8B) e dalla scelta di utilizzare la CPU o la GPU. L'utilizzo di una GPU sarebbe più efficiente ma si è scelto di limitare l'utilizzo alla CPU in quanto è l'unico hardware sicuramente disponibile, come descritto nella sezione precedente, in particolare l'analisi è stata svolta su un Ryzen 7 7840HS.
La massima potenza assorbita della CPU Ryzen 7 7840HS è di $54W$ ed in tali condizioni riesce a produrre mediamente 15 caratteri al secondo (numero che varia molto in base alla dimensione della storia della chat). Da queste informazioni è possibile calcolare che il **modello consuma** $1mWh$ **(milli-Watt ora) per carattere generato**.
Supponendo che vengano effettuate 10 richieste al giorno da 10 installatori, e ognuna generi 350 caratteri, si stimano **35000 caratteri giornalieri** totali.
Ciò porta il consumo giornaliero dell'app a

$$1mWh/carattere \times 35000caratteri = 35Wh = 0.035kWh$$

Per calcolare l'impatto ambientale del chatbot è necessario stimare l'energia che utilizza e convertirla in grammi di CO_2 equivalente emessa $gCO2e$. Considerando che secondo ISPRA nel 2023 le emissioni di CO_2 per kWh in Italia sono state pari a $257.2gCO2e/kWh$ (fonte) si ottiene che ogni giorno l'app produce

$$0.035kWh \times 257.2gCO2e/kWh = 9gCO2e$$

che per un anno significa

$$365g \times 9gCO2e = 32859gCO2e = 3.2859kgCO2e$$

Possiamo dire che si tratta di un risultato molto modesto: per rendere l'idea si consideri che un albero assorbe in media tra i 20 e i 50 kg di CO_2 ogni anno secondo diverse fonti. Il consumo annuo è infatti paragonabile a un viaggio di circa $25km$ con una utilitaria a benzina, come andare da Padova ad Arquà Petrarca con una Opel Corsa. Grazie all'obiettivo di funzionamento in locale e con risorse hardware limitare è stato possibile realizzare un prodotto leggero e dai consumi ridotti.

2 Riferimenti alla documentazione

Tutta la documentazione e i prodotti del gruppo PEBKAC sono disponibili presso:

https://pebkac-swe-group-11.github.io

Nello specifico i deliverables richiesti dal Proponente sono presenti nei seguenti documenti:

Deliverable	Documento	Vers.	Sezione
Documentazione sull'architettura realizzata	Specifica Tecnica	1.0.0	§3.1 Modello architetturale
Casi d'uso per l'utilizzo dell'applicativo web	Analisi dei Requisiti	2.0.0	§3 Casi d'uso
Documentazione di tutti i test realizzati e report, con relativo coverage e risultato	Piano di Qualifica	2.0.0	§4 Strategie di testing
Manuale di utilizzo dell'applicativo web per gli utenti	Manuale Utente	1.0.0	

Tabella 3: Riferimenti alla documentazione di progetto.

3 Codice sorgente

Repository contenente tutto il codice sorgente consultabile al seguente link:
`https://github.com/PEBKAC-SWE-Group-11/PEBKAC-GENIALE/tree/main/3%20-%20PB/mvp-src`

4 API

Documentazione delle API realizzare in formato OpenAPI (yaml) dispobibile al seguente link:
`https://github.com/PEBKAC-SWE-Group-11/PEBKAC-GENIALE/blob/main/3%20-%20PB/mvp-src/APIDocumentation.yaml`

5 Video dimostrativo

Un video che mostra l'applicazione in funzione è disponibile seguendo il link: