|  |  |
| --- | --- |
| ***PROJECT WORK***  ***L-31 Informatica per le aziende digitali*** | |
| **Cognome e Nome:** | Gigliotti Francesco |
| **Numero di Matricola**: | 0312200658 |
| **Corso di Studio:** |  |
|  |
|  |
| ◊   L-31 Informatica per le Aziende Digitali |
|  |
| **Tema n:** | 1 |
| **Titolo del tema:** | La digitalizzazione dell’ impresa |
| **Traccia del PW n:** | 4 |
| **Titolo della traccia:** | Sviluppo di una pagina web per un servizio di prenotazione online di un’ impresa del settore terziario |
| **Titolo dell’elaborato:** | Integrazione delle nuove tecnologie in ambito web e I.A. per la digitalizzazione delle imprese nel settore alberghiero. |
| **PARTE PRIMA – DESCRIZIONE DEL PROCESSO** | |
| **Utilizzo delle conoscenze e abilità derivate dal percorso di studio:** | Il progetto intende mettere in evidenza le competenze acquisite durante tutto il percorso di studio, in particolare sono state utili le conoscenze maturate nei corsi di:   * Basi Dati: per la progettazione e la creazione delle banche dati utilizzate nel progetto. * Reti di Calcolatori: per la conoscenza acquisita rispetto al livello Application dei modelli TCP/IP e ISO/OSI. * Metodologie di programmazione: per l’ apprendimento del paradigma di programmazione ad oggetti * Programmazione 2: che ha illustrato il funzionamento del linguaggio di programmazione Python * Tecnologie Web: fondamentale per la stesura del progetto, con questo corso è stato possibile approfondire i protocolli di comunicazione client-server, il funzionamento del linguaggio javascript, del DOM, Bootstrap e CSS, HTML e React. |
| **Fasi di lavoro e relativi tempi di implementazione per la predisposizione dell’elaborato:** | Per poter progettare, sviluppare il progetto è stato necessario suddividere il progetto in più fasi che potremmo descrivere come segue:   1. Analisi: in questa fase è stato necessario studiare il problema ed elaborare diverse soluzioni. Durante questa fase è stato fondamentale la stesura di una prima bozza del front-end, delle possibili chiamate del back-end e la realizzazione del diagramma ER, per il quale è stato utilizzato lo strumento MySql Model di MySql WorkBench che permette di creare il diagramma ER prima e automaticamente anche il DataBase vero e proprio. 2. Sviluppo: dopo la fase di analisi, si è passati allo sviluppo vero e proprio. Per tracciare gli avanzamenti dello sviluppo è stata creata una lavagna virtuale con trello dove sono stati suddivisi per task i servizi con al loro interno la lista delle chiamate back-end e dei componenti react da sviluppare. 3. Testing: terminato lo sviluppo di una chiamata o un componente questo viene testato per verificare il funzionamento regolare e corretto del componente. 4. Pushing: Per la salvaguardia del progetto e un ulteriore tracciabilità del progetto è stata usata una repository github. 5. Reiterazione passaggi 2 – 4 fino al completamento del progetto. |
|  |
|  |
|  |
|  |
| **Risorse e strumenti impiegati:** | Il progetto ha richiesto diverse fonti e strumenti per lo sviluppo.  Architettura:  L’architettura utilizzata per il progetto è stata una architettura a micto servizi. Questa scelta è stata effettuata sulla base dell’ alta scalibilità e della pulizia del codice. Per l’elevata fault-tollerance che consente ad un servizio di rimanere attivo anche se un’ altro servizio ha problemi.  Per il data base è stata scelto un’ approccio one schema for each service. Questo approccio è stato scelto per la facilità di integrazione è di sviluppo.  Gli strumenti utilizzati sono stati:   * Conda - per la gestione dei pacchetti python e la possibilità di creare un enviroment personalizzato per il progetto. * PyCharm - come IDE per python. Scelto per le molteplici funzioni di assistenza allo sviluppo. * VSCode – come text editor per lo sviluppo in react. Grazie ai suoi molteplici plug-in è uno strumento potente che riesce ad aiutare il programmatore a snellire il processo di sviluppo * MySql Workbench – è stato utilizzato come Data Base tool per la creazione e lo sviluppo del Data Base. Tra i suoi più grandi pregi c’è sicuramente la possibilità di creare e gestire il database partendo dal diagramma ER. * Fork – Git client che offre un’ interfaccia grafica user-friendly per la gestione delle repository git. * GitHub – utilizzato per il salvataggio del progetto e per la gestione delle vare feature e fix durante lo sviluppo. * ChatGPT – Questo Large Language Model è sicuramente il più conosciuto e popolare LLM al mondo. Grazie a questo strumento è stato possibile risolvere problemi e automatizzare parti dello sviluppo meccaniche e ripetitive come la creazione delle interfacce React agli oggetti ricevuti dal Back-end. * Claude – Un’altro LLM utilizzato come ChatGPT, essendo questi modelli generativi addestrati su dati differenti può essere utile vedere differenti approcci alla risoluzione di problemi per poter scegliere la più consona alla situazione. * StackOverflow – insieme ad altri forum come Reddit è estremamente utile per la condivisione di esperienze tra diveri programmatori. Essendo forum frequentati da persone reali. Le soluzioni qui riportate sono più affidabili che non su qualunque LLM. * YopMail – servizio per la creazione di mail temporanee utilizzate per effettuare utenze di test per la login. * FastApi – questo framework è stato sviluppato per efficientare al massimo le performace di Python. Grazie ad esso è semplice creare chiamate Rest e implementa di default OpenAPI per la gestione delle API lato server senza l’utilizzo di software terzi. * Postman – utilizzato raramente visto la presenza di Open API, ma utile in alcune circostanze, come ad esempio le chiamate con header personalizzato. * SQLAlchemy – un ORM molto popolare per la comunicazione tra python e il DataBase * MySQL – Un Database relazionale molto versatile e semplice da utilizzare. Oltre allo sviluppo su MySQL Workbench, MySql è stato scelto per la vasta community e la semplicità di trovare documentazione e nozioni online. * React – Per il Front end è stato utilizzato react come libreria. Sicuramente grazie alla sua ampia community e semplicità di utilizzo è una valida scelta moderna per lo sviluppo di un Front End. Si è scelto l’utilizzo di typescript anziché il classico JavaScript essendo typescript un’estenzione di quest’ultimo. * Python – Sebbene sia un liguaggio interpretato e debolmente tipizzato. Python è famoso, e apprezzato, per la velocità di sviluppo. * Bootstrap – una famoso Framework CSS per gli stili. Open Source, aggiornato e con una vasta community alle spalle, rendono bootstrap un ottima scelta tecnologica * VITE – Un server di sviluppo front-ent molto conosciuto. * Npm – usato come package manager per react. * LangChain – Un famoso framework open source per la creazione di agenti AI e RAG. Utilizzato, in questo caso, per creare un chatbot di supporto utilizzando un RAG con la documentazione testuale in inglese del sito. * Ollama – Un Ai model manager, anch’ esso Open Source. Utilizzato sia per il chatbot di assistenza che per l’embedder per il db vettoriale. * Llama 3.2 – Un LLM opensource sviluppato da Meta, compatibile con ollama. La versione utilizzata è quella da 3 milirdi di parametri, scelta in quanto abbastanza piccola ma più grande della versione da 1 miliardo di parametri che purtroppo non restituisce risultati accettabili per il task richiesto. * Nomic – Un text ebbedder compatibile con ollama. Utilizzato per vettorializzare il testo della documentazione e salvarlo in Chroma DB. * Chroma – Un database vettoriale, utilizzato per salvare la documentazione usata per il chatbot.   Altre Librerie Python:  pip install pyjwt  pip install PyMySQL  pip install fastapi  pip install SQLAlchemy  pip install starlette  pip install pydantic  pip install typing  pip install uvicorn  pip install passlib[bcrypt]  pip install PyJWT  pip install python-multipart  pip install fastapi-mail  pip install httpx  pip install canvas  pip install reportlab  pip install langchain  pip install langchain-community  pip install langchain\_chroma  pip install chromadb  pip install langchain-ollama  Fonti e Risorse:   * https://fastapi.tiangolo.com/ * https://medium.com/@amirm.lavasani/how-to-structure-your-fastapi-projects-0219a6600a8f * https://docs.sqlalchemy.org/en/20/ * <https://getbootstrap.com/docs/5.3/getting-started/introduction/> * <https://python.langchain.com/docs/introduction/> * https://youtu.be/yF9kGESAi3M?si=59Mu039j8Jwmk6xP&t=4578 |
| I motivi delle mie scelte sono stati riportati sopra |
| Ho reperito la maggior parte del materiale da internet. Grazie a forum, video youtube, consigli dai sopracitati LLM e conoscenze professionali; |
| Le più grandi difficoltà sono state riscontrate nella configurazione del progetto. Purtroppo la documentazione fornita da Fast API per l’organizzazione dei servizi è scarsa, grazie all’ articolo di Medium riportato tra le fonti sono riuscito ad organizzare un template per la creazione dei progetti.  Una volta organizzata la struttura dei progetti ho riscontrato un problema con la lettura dei file dentro di essi. Stavolta, grazie a StackOverflow, sono riuscito a capire che la working directory di default di pithon non mi consentiva la lettura dei file esterni. Modificando questa impostazione dal’ edit configuration di PyCharm sono riuscito a risolvere il problema.  Un nuovo grande scoglio è stata l’impostazione e la creazione del servizio dedicato al RAG. Questa volta ho dovuto seguire un tutorial su youtube per creare un primo dummy project per imparare ad usare le varie tecnologie (il tutorial in questione è riportato tra le fonti utilizzate). Non volendo utilizzare API di LLM a pagamento, ho modificato il codice per poter eseguire il tutto in locale tramite LLM openSource utilizzando llama 3.2 da 3 miliardi di parametri, che dalle prove effettuate è risultato essere il modello più piccolo della famiglia llama per ottenere un risultato accettabile. Purtroppo non sono riuscito a gestire correttamente la sostituzione del database vettoriale, salvato localmente nel progetto. Pertanto, qual’ ora fosse necessaria la cancellazione del database vettoriale, sarebbe necessario effettuare tale operazione a server spento.  Inoltre per questioni temporali non sono riuscito ad implementare lato back-end e front-end ADMIN, la parte relativa alle review. Le parti implementate lato front-end USER sono solo relative alla visualizzazione solamente delle suddette, utilizzate come concept. |
| **PARTE SECONDA – PREDISPOSIZIONE DELL’ELABORATO** | |
| **Obiettivi dell’elaborato/progetto/artefatto:** | L'obiettivo principale del progetto è sviluppare una piattaforma innovativa per la prenotazione di hotel, caratterizzata da semplicità d’uso e interfaccia intuitiva, che consenta agli utenti di consultare e comparare in un unico portale diverse opzioni alberghiere. La piattaforma deve garantire una navigazione fluida e offrire strumenti efficienti per scegliere l’hotel più adatto alle proprie esigenze, basandosi sui seguenti criteri:   1. **Prezzo**: Consentire agli utenti di filtrare e ordinare gli hotel in base alle proprie disponibilità economiche. 2. **Locazione**: Fornire strumenti di ricerca geolocalizzata per individuare strutture situate in zone di interesse specifico. 3. **Servizi offerti**: Offrire la possibilità di consultare dettagli su servizi aggiuntivi (es. Wi-Fi, parcheggio, accessibilità) e valutarne la pertinenza rispetto alle necessità dell’utente.   Inoltre, il progetto include lo sviluppo e l'integrazione di un sistema di **assistenza basato sull’intelligenza artificiale**, operativo 24 ore su 24, 7 giorni su 7. Questo servizio sarà progettato per rispondere alle domande più comuni degli utenti, garantendo così un supporto immediato e migliorando l’esperienza utente complessiva.  Un ulteriore obiettivo è assicurare che la piattaforma sia progettata per garantire:   * **Facilità di gestione**: Un’interfaccia backend intuitiva per consentire aggiornamenti frequenti e una gestione semplice delle informazioni sugli hotel. * **Scalabilità**: Una struttura modulare e scalabile per supportare un numero crescente di utenti e hotel, senza comprometterne le prestazioni. * **Manutenibilità**: Codice strutturato e ben documentato per facilitare interventi di manutenzione e aggiornamenti futuri.   Gli obiettivi del progetto riflettono un approccio focalizzato sull’utente, sull’innovazione tecnologica e sulla sostenibilità dello sviluppo nel tempo, in linea con le migliori pratiche di progettazione software e ingegneria dei sistemi. |
| **Contestualizzazione:** | Contesto teorico Il modello operativo del progetto si basa su principi di e-commerce e marketplace digitali, dove domanda e offerta vengono messe in contatto in modo diretto.Questo approccio sfrutta:   * **Economia della piattaforma**, che massimizza il valore tramite l’interconnessione di utenti e fornitori. * **User Experience (UX)** per rendere semplice e accogliente la navigazione all’ interno dell’applicazione * **LLM** per una prima risposta alle domande più semplici degli utenti  Contesto applicativo Questo progetto consente agli utenti di:   * Prenotare strutture alberghiere e non, con opzioni di cancellazione flessibili. * Personalizzare la ricerca con filtri avanzati. * Accedere a tariffe competitive grazie alla contrattazione diretta con i partner. |
| **Descrizione dei principali aspetti progettuali:** | **Front-End:**  **Lato Utente:**  **Barra di Navigazione (NavBar)**    La **NavBar** è una striscia scura situata nella parte superiore di ogni pagina web. Include due componenti principali:   1. **Hotel**: Reindirizza l'utente alla homepage. 2. **Utente**: Questo componente si aggiorna dinamicamente in base all'autenticazione dell'utente:    * Se l'utente non è connesso, visualizza "Sign In".    * Se l'utente è connesso, mostra il nome utente dell'account collegato.  Footer   Il **Footer** si trova sempre nella parte inferiore di ogni pagina web. Contiene le **informazioni di contatto** del sito, come il numero di telefono, l'email, Facebook, Twitter e LinkedIn. Per assistenza o supporto, gli utenti possono fare riferimento a questa sezione per i contatti. Homepage   La **Homepage** è la prima pagina visualizzata accedendo al sito. È composta da tre sezioni principali:   1. **NavBar**: In alto. 2. **Footer**: In basso. 3. **Modulo di Prenotazione**: Posizionato al centro della pagina.  Modulo di Prenotazione    * **Campo Città**: Permette agli utenti di cercare una città in cui prenotare un hotel. Mentre si digita, viene mostrata una lista di città corrispondenti. Se non compaiono risultati, significa che non ci sono hotel disponibili in quella città. * **Date di Check-In e Check-Out**: Specificare le date di arrivo e di partenza per la prenotazione. * **Numero di Ospiti**:   + **Adulti**: Per gli ospiti di età pari o superiore a 12 anni.   + **Bambini**: Per gli ospiti di età inferiore a 12 anni.   Dopo aver cliccato sul pulsante **Cerca**, gli utenti verranno reindirizzati alla pagina **Booking Label**. Accedi (Log In)   Per accedere, gli utenti possono cliccare su **Sign In** nella NavBar. Ci sono due modalità di accesso:   1. Inserire il nome utente/email e la password. 2. Utilizzare un **Codice di Accesso (Sign-In Code)**.  Password Dimenticata Cliccare su **Forgot Password?** per aprire un modulo che invia un codice di accesso per reimpostare la password. Registrati Se gli utenti non hanno un account, possono crearne uno cliccando su **Sign Up** nella sezione sotto "Forgot Password?".  I dati richiesti sono:   * Nome * Cognome * Numero di Telefono * Nome Utente * Password * Indirizzo Email   Dopo aver completato il modulo e cliccato sul pulsante **Sign Up**, il sistema creerà l'account. Utilizza un Codice di Accesso Per accedere con un **Codice di Accesso (Sign-In Code)**:     1. Inserire il nome utente o l'email. 2. Cliccare sul pulsante **Send Code**. Un'email con il codice verrà inviata (valida per 5 minuti). Se non si riceve l'email, controllare la cartella spam o cliccare su **Retry**. 3. Inserire il codice e cliccare su **Verify** per accedere.  Booking Label La pagina **Booking Label** mostra le camere disponibili in base ai filtri di ricerca.   Funzionalità:  * **Modulo Filtri**: Posizionato a sinistra per modificare i parametri di ricerca. * **Visualizzazione Camere**: Elenca le camere disponibili o mostra un messaggio ("No rooms available") se nessuna corrisponde ai criteri.   Per le camere disponibili, i dettagli includono:   * Nome dell'hotel, numero di stelle e posizione. * Tipo di camera (es. Suite, Singola, Doppia). * Numero di letti e ospiti. * Costo per notte (per adulti e bambini sotto i 12 anni). * Costo totale e recensioni degli utenti.   Un pulsante **Details** reindirizza alla pagina **Room Detail** per maggiori informazioni. Dettaglio Camera   La pagina **Dettaglio Camera** fornisce:   * Nome dell'hotel, numero di stelle e posizione. * Descrizione della camera e servizi offerti. * Informazioni sulla rimborsabilità.  Prenotare una Camera Cliccare su **Book Now** per procedere con la prenotazione. Il modulo di prenotazione richiede:   * Numero di carta * Data di scadenza * CVV   Dopo il pagamento, apparirà una finestra di conferma e verrà inviata un'email di conferma.  Chiaramente questa parte in realtà non è collegata ad alcun servizio back-end. Area Privata L'**Area Privata** è accessibile cliccando sul nome utente nella NavBar (per utenti connessi). Funzionalità:  1. **Account**: Gestire le informazioni personali. 2. **Reset Password**: Aggiornare la password utilizzando un modulo simile alla sezione "Password Dimenticata". 3. **Reservations**: Visualizzare le prenotazioni attive. 4. **My Reviews**: Visualizzare le recensioni inviate. 5. **Log Out**: Effettuare il logout dall'account. 6. **AI FAQ Assistant Bot (Botto)**: Fornisce risposte alle domande sul sito.             **Lato Amministratore:**  Effettuando l’accesso con un profilo ADMIN è possibile accedere alla console, che permette di aggiungere hotel, servizi, caricare i template per le mail, aggiornare il documento per ChatBot, ecc.  Attenzione:  A differenza del profilo utente normale, un profilo amministratore può essere creato solo da un’ altro profilo amministratore. Ciò implica che il primo profilo amministratore, ovvero al primo avvio dell’ applicazione con il database vuoto, debba essere creato o direttamente dal data base o tramite API.  La **Homepage** è la prima pagina visualizzata accedendo alla console. È composta da tre sezioni principali:   1. NavBar: Dove questa volta sono presenti i nomi delle Entità da modificare 2. Footer: Nella console il footer è costituito solamente dal nome dell’ applicazione e dall’ anno del copyright 3. Centro gestione Hotel: la prima pagina che incontriamo nella console, non che home page, è la modale per la gestione degli hotel     **Hotel Management**:  Nella Modale per la gestione degli hotel è possibile:   * Aggiungere un nuovo hotel tramite il pulsante “Add New” * Modificare i dati dell’ hotel tramite il pulsante “Edit” * Cancellare un hotel tramite il pulsante “Delete” * Gestire le stanze dell’ hotel tramite il pulsante “Rooms”   **Edit e Add New**  I pulsanti “Edit” e “Add New” aprono la stessa modale con l’unico cambiamento che nel caso del pulsante “Edit” i campi del form verranno riempiti con i dati dell’ hotel che si vuole modificare, mentre se si clicca il bottone “Add New” i campi saranno vuoti.  **Cancellazione**  Premendo il pulsante “Delete” la riga dell’ hotel selezionato sarà cancellata, e a cascata anche ogni stanza appartenente all’ hotel cancellato.  **Rooms**  La sezione Rooms serve per gestire, appunto, le stanze di un hotel. Cliccando l’omonimo tasto si aprirà la seguente modale:  Dove cliccando ”Add Room” sarà possibile aggiungere una nuova stanza da associare all’ hotel precedentemente scelto.  **“Room Detail”**, invece consente di modificare le caratteristiche di una stanza e/o aggiungere eventuali servizi e immagini della stanza.  Per le foto e i servizi le modali sono simili e i comportamenti per attivarle sono gli stessi:   * tramite il click sull’ icona + è possibile aggiungere una foto/servizio * tramite un click sulle icone dei servizi o su una foto della stanza è possibile modificarle   Per aggiungere o modificare la foto di una stanza bisogna incollare un url di una foto. Mentre per i servizi si possono selezionare i servizi da un menù a tendina dove sono elencati tutti i servizi disponibili.  Generalmente tutte le altre pagine della console hanno la stessa struttura a modali, pertanto per ragioni di spazio non andrò a mostrarle.  **Data Base:**  Per il database è stato utilizzato un approccio schema for each service, utilizzando uno schema differente per ogni servizio.  Ogni rettangolo è uno schema.  Per vedere nel dettaglio gli schema, è possibile entrare nella repository del progetto ed entrare nella directory db-schema dove sono presenti i file .mwb che consentono tramite MySQL Workbench di creare il db dal diagramma ER.  **Back-End:**  Tutti i servizi back-end sono strutturati come da immagine.  Una cartella, denominata come il nome del servizio, contiene una cartella src e un file di enviroment.  Nel file di enviroment sono presenti dati sensibili come l’url daccesso al DB e gli ip delle macchine dei server su cui girano i servizi con cui il servizio deve interagire.  Nella cartella src, utilizzata come source path per ogni progetto, sono presenti varie sottocartelle utilizzate per delle classi con compiti specifici. Ad esempio, nella cartella config, avremo la classe che si occupa di connettersi al db e la classe che recupera i dati presenti nel file di enviroment. Nella cartella routers ci saranno i controller dell’applicazione. In Models ci saranno i modelli che si interfacceranno con il DB, mentre in schemas gli oggetti di input e di output. Nella cartella repositories ci sanno tutti i metodi che si interfacciano con il DB, mentre nella cartella services ci saranno i metodi che compiono operazioni sui dati ottenuti dai controller e/o dal db. Infine in utils ci saranno tutti i metodi creati appositamente per aiutare gli sviluppatori.  Il file di main consente di far partire l’applicativo utilizzando uvicorn e qui sono specificati i permessi di accesso al servizio CORS.    Tutti i servizi sono veramente simili tra loro e molto semplici.  Vorrei pertanto concentrarmi sul servizio per il chatbot, il quale è molto differente dagli .  **AI\_service**  Per realizzare il servizio sono state usate tecnologie totalmente opensource, dal modello di embedding al vero e proprio LLM utilizzato.  Per far funzionare correttamente il servizio è necessario l’installazione di ollama sul proprio PC.  Dopo l’installazione di ollama è necessario tramite esso l’installazione di nomic, che funzionerà da embedder per il db vettoriale, e di llama3.2:3b (il modello da 1 solo miliardo di parametri è stato testato ma con risultati non soddisfacenti) o llama3.1 (personalmente è stato testato solo il modello da 8 miliardi di parametri, ma penso che vada bene un qualsiasi modello della versione 3.1). chiaramente si posso utilizzare anche altri modelli, ma potrebbe richiedere il cambiamento di alcuni prompt utilizzati.  **vectorized\_db\_manager.py**  vectorized\_db\_manager.py si occupa di gestire il database vettoriale (Chroma) per salvare le informazioni fornite dal file fornito dagli amministratori.  I file utilizzato non è che una documentazione testuale informale del sito lato utente. Grazie alla API dedicate è possibile fare un upload del file e inizializzare il database.  La funzione di inizializzazione non fa che caricare il file specificato tramite la classe TextLoader fornita da lang chain, effettuare uno split del testo tramite la classe CustomTextSplitter creata appositamente per suddividere il testo utilizzando un pattern da me specificato (in questo caso “\n---\n” usato come separatore nel file “text\_polished\_documentation\_en.txt”). Successivamente tramite la classe Chroma, fornita sempre dal framework lang chain, vengono passati sia l’embedder, ottenuto dalla classe di lang chain OllamaEmbeddings, che il path della directory dove si vuole creare il database vettoriale.  Una volta fatto ciò si può passare al database, tramite il metodo from\_documents(), il file suddiviso in chunks, l’embedder e il path della directory di salvataggio del database vettoriale e creare finalmente il database.  **ai\_service.py**  ai\_service.py si occupa invece di interrogare il db e ottenere il corrette informazioni dal database vettoriale.  La funzione che si occupa di interrogare il database e resttuire il valore corretto è retrieve\_llm\_response(), che dato in input una domanda di un utente, torna la risposta rielaborata dal llm, ma solo se presente all’interno del file precedentemente caricato.    Per prima cosa il metodo controlla se effettivamente esiste il database, se esiste continua l’esecuzione, altrimenti lancia un’ eccezione con un messaggio che avverte l’utente di creare prima un database vettoriale.  Una volta accertatosi che il database sia effettivamente presente all’ interno del server, vi accede settando un embedder e il path del database. Successivamente viene creato un retriever che recupererà i primi tre risultati più simili data una certa soglia, qui settata a 0.1.  Viene poi settato il llm.  Una volta creato il retriever e impostato il llm da utilizzare, possiamo fornire un prompt di contesto e il prompt per la restituzione delle risposte, così definiti:  Una volta definiti questi prompt possiamo creare le catene che ci serviranno per far funzionare il nostro rag.  Lang chain è un framework che si basa sulla creazione di queste catene, che sono così definite dalla documentazione ufficiale di lang chain: “Le **catene** si riferiscono a sequenze di chiamate, che possono essere rivolte a un LLM (modello linguistico di grandi dimensioni), a uno strumento o a un passaggio di pre-elaborazione dei dati.”  Vengono quindi create le due catene history\_aware\_retriever e question\_answare\_chain, che verranno concatenate tra di loro per creare la catena per il rag, ovvero rag\_chain.  Una volta creata la catena per il rag, per utilizzarla e data in input una query otternere la risposta desiderata, basterà semplicemente chiamare il metodo invoke e restituire il risultato:    Repository git: https://github.com/xFranco99/thesis-microservice-hotel-booking-monorepo.git |
| **Campi di applicazione:** | Il progetto si colloca nell’ambito del turismo digitale, con l’obiettivo di semplificare la ricerca e la selezione di hotel per i turisti. La piattaforma proposta offre un sistema centralizzato per la consultazione e il confronto di diverse strutture alberghiere, rendendo il processo decisionale più rapido, informato e personalizzabile in base alle esigenze dell’utente.  Dal punto di vista del mercato, l’iniziativa favorisce la concorrenza tra gli albergatori, incentivandoli a migliorare la qualità dei servizi offerti per ottenere valutazioni migliori e attirare una clientela più ampia. Questo meccanismo di feedback diretto da parte degli utenti può contribuire a un’evoluzione positiva del settore alberghiero, con benefici sia per i consumatori sia per i fornitori di servizi.  Inoltre, il progetto mira a ridurre i costi operativi per le strutture alberghiere. La piattaforma rappresenta una soluzione economica e accessibile rispetto alla creazione e gestione di un portale web dedicato per ciascun hotel. Grazie alla centralizzazione dei servizi e alla condivisione delle infrastrutture digitali, gli albergatori possono concentrarsi sul miglioramento delle loro offerte senza dover sostenere gli oneri di sviluppo e manutenzione di un sito web personalizzato. |
| **Valutazione dei risultati (potenzialità e criticità):** | Il progetto, sebbene completato in parte e operativo per l’utilizzo con dati di test, presenta alcune criticità e limiti legati principalmente ai ristretti tempi di realizzazione disponibili. Questi aspetti limitano il pieno raggiungimento degli obiettivi prefissati e sottolineano aree che richiedono ulteriori sviluppi e miglioramenti. Di seguito, vengono analizzati i principali punti critici:   1. **Mancanza del sistema di review**: Una delle funzionalità chiave per garantire una maggiore trasparenza e supportare gli utenti nelle loro decisioni, ovvero il sistema di recensioni, non è stata implementata. L’assenza di questa componente riduce la capacità della piattaforma di favorire una valutazione collaborativa e di incentivare gli albergatori a migliorare i propri servizi in base al feedback ricevuto. 2. **Design pattern non applicati correttamente**: Il progetto non ha implementato appieno i design pattern ideali per un’architettura basata su microservizi. Un possibile esempio potrebbe essere il pattern **CQRS** (Command Query Responsibility Segregation), fondamentale per separare le operazioni di lettura da quelle di scrittura e migliorare le performance e la scalabilità del sistema, non è stato adottato. Questa mancanza può limitare la capacità del sistema di gestire un aumento del carico e di rispondere in maniera efficiente a richieste concorrenti. 3. **Gestione e implementazione delle chiamate asincrone**: La gestione delle chiamate asincrone, fondamentale per garantire una comunicazione efficiente tra front-end e back-end, non è stata ottimizzata. L’implementazione incompleta o non conforme agli standard potrebbe causare ritardi nell’elaborazione delle richieste, comportare problemi di sincronizzazione dei dati e ridurre l’efficienza complessiva della piattaforma. Penso in particolar modo al servizio per il chatbot. 4. **Gestione incompleta della sessione utente lato front-end**: Non è stato implementato un sistema per il controllo dello stato delle sessioni utente, ad esempio per rilevare una sessione scaduta. Ciò potrebbe portare a problematiche legate alla sicurezza, come l'accesso non autorizzato, o compromettere l’esperienza utente, causando possibili errori nel flusso di navigazione. 5. **Comunicazione tra servizi tramite messaggistica**: Non è stata implementata una soluzione basata su sistemi di messaggistica, come **Kafka queue**, per garantire una comunicazione affidabile, scalabile e asincrona tra i microservizi. L’assenza di un middleware per la gestione delle code riduce la capacità della piattaforma di gestire processi complessi o elevati volumi di richieste, oltre a compromettere la resilienza del sistema in caso di guasti o ritardi.  Potenziali impatti delle criticità  * **Esperienza utente limitata**: L’assenza di funzionalità chiave come il sistema di recensioni e la gestione delle sessioni può influire negativamente sulla percezione complessiva della piattaforma, riducendo la fiducia e l’engagement degli utenti. * **Scalabilità e manutenibilità compromesse**: La mancata adozione di design pattern appropriati per i microservizi potrebbe causare difficoltà nella gestione di un aumento delle richieste o nell’implementazione di nuove funzionalità. * **Sicurezza e affidabilità**: La gestione incompleta delle sessioni può esporre la piattaforma a vulnerabilità, riducendo la robustezza del sistema in scenari reali.   Questi limiti, sebbene comprensibili nel contesto delle risorse e dei tempi a disposizione, evidenziano aree prioritarie per sviluppi futuri, con l’obiettivo di migliorare la qualità e la sostenibilità del progetto. |