Report progetto CultureLLM

Fabio Falconi
Matricola / 2048079
La Sapienza Università di Roma

Gabriele Lapini
Matricola / 2054678
La Sapienza Università di Roma

1 Obiettivo del progetto

Il presente progetto riguarda lo sviluppo di un'applicazione Web che permetta agli utenti di porre domande sulla cultura italiana ricevendo risposte da intelligenze artificiali e altri utenti, per poi dover scegliere tra le risposte ricevute quella che ritengono migliore e individuare quale tra di esse è stata data da un altro utente. Le azioni degli utenti sono caratterizzate dall'assegnazione di punti che vengono mostrati in una classifica in tempo reale, per stimolare gli utenti ad interagire maggiormente. I dati raccolti con questa applicazione possono fornire interessanti informazioni sulla qualità delle risposte IA, cosa gli utenti considerano come una buona risposta e sulla difficoltà nel distinguere risposte umane da quelle artificiali.

2 Struttura del progetto

Il sistema è strutturato in quattro servizi interdipendenti: il Database, il Backend, il Frontend e l'IA. L'interazione tra questi servizi è illustrata nella Figura 1.

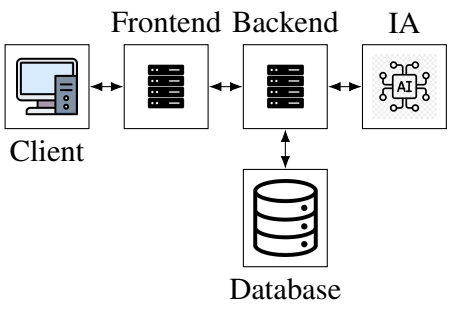

Figure 1: Struttura complessiva

3 Organizzazione del database

Nell'ambito della modellazione concettuale del sistema, si è scelto di identificare quattro entità distinte: User, Theme, Question, Answer.

User è la classe dell'utente, e contiene nei suoi attributi tutti i suoi dati: ID (PK[1]), username, password, score (il punteggio dell'utente), friendcode (un codice casuale a 16 cifre univoco per utente, che può essere usato da amici dell'utente per registrarsi e avere un bonus di punti. Serve ad incentivare nuovi utenti a registrarsi).

Theme è la classe del tema. Ogni domanda deve essere associata ad uno degli elementi di questa classe. Ha come attributi: ID (PK), theme (nome del tema).

Question è la classe della domanda, ogni domanda posta fa parte di questa classe. Ha come attributi: ID (PK), payload (la domanda vera e propria), answered, checked, is_answering (tre campi che servono nel codice rispettivamente a riconoscere se la domanda ha ricevuto risposta da un utente, le sue risposte sono state valutate dall'autore e quale utente sta rispondendo a quella domanda in quel momento).

Answer è la classe delle risposte, tutte le risposte di utenti e IA fanno parte di questa classe. Ha come attributi: ID (PK), payload (la risposta vera e propria), best (booleano che indica se un utente l'ha scelta come risposta migliore per la domanda)

Le relazioni tra le entità sono le seguenti:

Argument: Una relazione di tipo uno-a-molti che connette Question e Theme.

Asked: Una relazione di tipo uno-a-molti tra Question e User.

Answered: Una relazione di tipo uno-a-molti che connette Answer e User.

Reply: Una relazione di tipo uno-a-molti tra Question e Answer.

La rappresentazione schematica di tali entità e relazioni, con le rispettive cardinalità, è illustrata in Figura 2.

[1]PK - Primary key FK - Foreign key

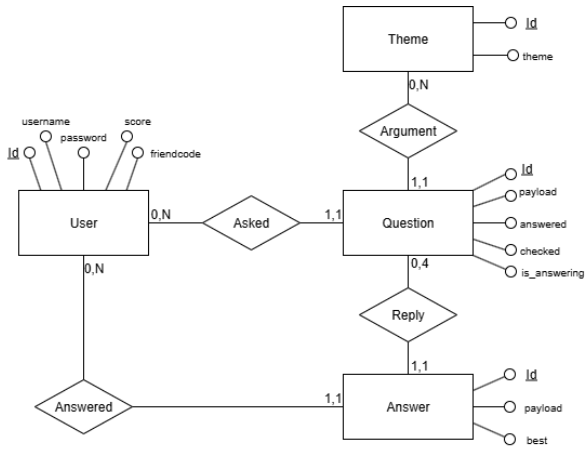

Figure 2: Schema concettuale Entità-Relazione

Secondo i principi della traduzione logica, ciascuna entità e relazione identificata nello schema concettuale viene mappata su una tabella distinta nello schema relazionale. Tuttavia, le relazioni di tipo uno-a-molti vengono incorporate nell'entità con cardinalità 'uno' nella suddetta relazione (quindi, per esempio, la relazione Argument sarà incorporata nella classe Question).

Pertanto, la traduzione logica del modello concettuale precedentemente descritto produce le seguenti tabelle: User (ID - PK, username, password, score, friendcode), Theme (ID - PK, theme), Question (ID -PK, payload, answered, checked, is_answering, author - FK verso User che ingloba la relazione Asked, theme - FK che ingloba la relazione Argument), Answer (ID - PK, payload, best, author - FK che ingloba la relazione Answered, question - FK che ingloba la relazione Reply)

Per evitare che alcune domande si ripetano troppo spesso e avere dei dati più vari, il payload di Question è stato impostato come Unique. Lo stesso vale per Username e Friendcode di Users, che vogliamo siano unici per ogni utente.

Lo schema relazionale risultante è illustrato nella Figura 3.

3.1 Popolamento iniziale

All'avvio, nel database viene inserito un utente con ID=-1, che rappresenta l'IA, necessario per permettere l'inserimento di domande che non siano associate ad alcun utente, vista la foreign key del campo author di Question verso ID di User. Vengono poi inseriti i temi delle domande tra cui gli utenti potranno scegliere, e alcune domande iniziali associate all'utente IA, per permettere al primo avvio comunque agli utenti sia di porre domande sia di rispondere.

4 Organizzazione del codice

4.1 Backend

Il servizio di Backend implementa la logica applicativa del sistema e la comunicazione con database e IA, ed è composto dai seguenti moduli con le rispettive responsabilità:

models.py, database_connection.py: Definizione delle strutture dati interne di input/output degli altri moduli e della classe per la gestione della connection pool con il Database.

execute_query.py: Gestione centralizzata delle operazioni di interazione con il database (interrogazioni e modifiche). Tutte le comunicazioni con il database da parte degli altri moduli del Backend sono mediate dalle funzioni qui definite, garantendo un controllo unificato della gestione degli errori. Ogni richiesta avviene tramite l'esecuzione di prepared statement, per evitare SQL injection.

ai_answers.py: Implementazione della logica di richiesta di risposte all'IA e del suo inserimento nel database.[2]

[2]Per questa consegna è stata usata come IA ollama con il modello gemma3:1b, quando disponibile si sostituirà con ia personalizzate e più performanti

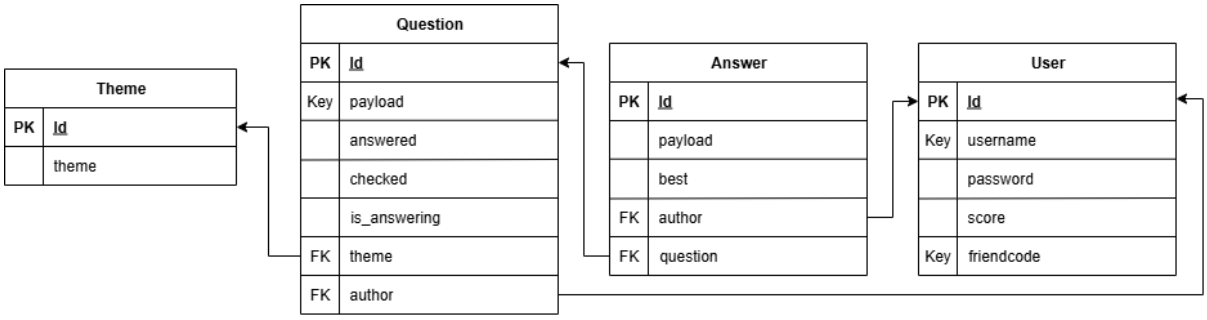

Figure 3: Schema Relazionale risultante

backend.py: Modulo principale del servizio Backend. Si occupa della definzione e gestione degli endpoint dell'API utilizando il framework FastAPI. Le funzionalità esposte dagli endpoint includono:

- **/health:** Endpoint per il monitoraggio dello stato del servizio.
- **/login:** Endpoint per il controllo della presenza dell'utente nel database, e in caso di successo imposta un cookie nel browser dell'utente per riconoscerlo nelle successive richieste. Non permette l'accesso con l'utente con ID=-1.
- **/register:** Endpoint per l'inserimento di un nuovo utente nel database, con tutti i suoi dati, dopo aver verificato che non è presente già un utente con lo stesso username
- **/ask:** Uno degli Endpoint printipali dell'applicazione. Riceve una domanda dall'utente, e si occupa del controllo, tramite IA, che la domanda posta sia coerente col tema scelto. In caso affermativo, inserisce la domanda nel database, assegna i punti all'utente, e avvia un thread dove l'IA genera le sue risposte alla domanda appena inserita, e le inserisce nel database. L'endpoint poi restituisce tutti i dati aggiornati da dover mostrare nella UI. Dai temi che restituisce all'utente per la nuova domanda, rimuove temporaneamente il tema della domanda appena posta, per stimolare la richiesta di domande più varie.
- **/validate:** Endpoint chiamato quando nella UI si seleziona di voler rispondere ad una domanda, quando ha ricevuto tutte le risposte. Si occupa di fornire tutti i dati necessari per quella domanda, in particolare le sue risposte.
- **/best e /human:** Endpoint per gestire quale sia la scelta della risposta che l'utente ritiene sia la migliore, o ritiene sia data da un altro utente. Dopo la chiamata a /validate, le scelte degli utenti chiamano questi due endpoint rispettivamente quando scelgono la risposta migliore e la risposta che ritengono sia di un utente. Si occupano di memorizzare le scelte e assegnare punti quando necessario.
- **/answer:** Un altro degli endpoint principali dell'appicazione. Riceve una risposta con l'id della domanda cui si riferisce e se la risposta non è vuota si occupa di inserirla nel database, con tutte le sue informazioni. Poi restituisce una nuova domanda a cui l'utente possa rispondere.
- **/leaderboard:** Endpoint che si occupa di restituire una lista completa degli utenti con i loro punteggi, da mostrare nella classifica in tempo reale. Non include le informazioni sull'utente con ID=-1.
- **/profile:** Endpoint che si occupa di restituire informazioni complete sull'utente che ha effettuato la richiesta, tra cui i suoi dati e le sue statistiche, ma non la sua password, che deve rimanere nascosta.
- **/passreset:** Endpoint che si occupa di modificare la password dell'utente. Riceve la nuova password ripetuta due volte. Se le due password coincidono, vengono impostate nel database per l'utente che ha effettuato la richiesta.
- **/check_new_answers:** Endpoint che si occupa di restituire informazioni complete su tutte le domande che devono ancora ricevere validazione e poste dall'utente che effettua la richiesta.
- **/logout:** Endpoint che si occupa di eliminare il cookie dal browser dell'utente e rimuovere le risorse che gli erano state assegnate (per esempio resettare il parametro is_answering della domanda associata all'utente).
- **Funzioni di Startup e Shutdown:** Funzioni designate per l'esecuzione all'avvio e alla chiusura dell'applicazione. Si occupano dell'instaurazione e della chiusura della connection pool col database e di chiamare il download del modello per l'ia, nel caso non sia presente.

db_utils.py, generic_utils.py, jxt_utils.py: Moduli che contengono funzioni ausiliarie per il backend. I tre moduli si occupano, rispettivamente, delle funzioni di concessione di una connessione dal pool all'endpoint che la richiede, delle funzioni di individuazione dell'utente dato il cookie ricevuto nella richiesta, e delle funzioni di codifica e decodifica dei cookie. Tutte queste funzioni vengono ampiamente usate nella maggior parte degli endpoint.

4.1.1 Assegnazione dei punti

Il sistema di punteggio è stato ideato in modo da essere bilanciato e conferire punteggi per le azioni di nostro interesse, in modo da stimolare gli utenti ad interagire. I punti viengono asssegnati in questo modo:

- **+50 punti:** Per aver inserito il codice amico alla registrazione. Questa scelta è stata adottata per incentivare degli utenti a partecipare se dei loro conoscenti già partecipano, poichè inserendo il codice dei loro conoscenti riceverebbero dei punti alla creazione del profilo. Tuttavia si è scelto di evitare, per il momento, di assegnare punti a colui che fornisce il codice, per evitare che un utente sfrutti questa possibilità per creare ripetutamente dei nuovi profili non utilizzati solo per ottenere punti sul proprio profilo principale.

- **+10 punti:** Per aver posto una domanda. Questa scelta serve per stimolare gli utenti a porre numerose domande. Il numero di domande massime in lavorazione è stato tuttavia posto a 10 per evitare che gli utenti pongano unicamente domande a ripetizione ma senza mai rispondere.

- **+40 punti:** Se una propria risposta viene scelta come "migliore" dall'utente che ha posto la domanda. Ricevere punti solo se la domanda viene scelta come migliore incentiva gli utenti a formulare domande complete e coerenti. La scelta di assegnare 40 punti deriva dal fatto che le risposte tra cui un utente sceglie la migliore sono 4, e considerando uguale probabilità che ognuna di esse venga scelta, il valore atteso dei punti che vengono guadagnati diventa +10, ovvero lo stesso punteggio di quando si pone una domanda.

- **+10 punti:** Per aver indovinato quale delle 4 risposte era stata data da un'altro utente. Per inserire un ulteriore elemento di gioco nell'applicazione.

4.2 Frontend

Il servizio di Frontend gestisce la comunicazione tra il Backend e l'utente, tramite interfaccia web HTML. Include infatti il modulo principale e i template HTML necessari, con associati i rispettivi moduli css e javascript:

frontend.py: Modulo principale del servizio Frontend. Svolge il ruolo di intermediario tramite un framework FastAPI. Gli endpoint definiti in questo modulo sono:

• /login • /register • /passreset • /ask • /validate • /best • /human • /answer • /profile • /check_new_answers • /logout

Sono endpoint che hanno la responsabilità di inoltrare le richieste dell'utente (e i relativi parametri) agli endpoint omonimi del Backend, per poi restituire al client il template HTML giusto con i parametri corretti.

• /get_tab_content/ask • /get_tab_content/answer • /get_tab_content/leaderboard

Sono endpoint che hanno la responsabilità di inoltrare le richieste dell'utente agli endpoint ask, answer e leaderboard del Backend, ma con dei parametri placehoder specifici che gli permettano di ottenere unicamente i dati da restituire al client nel template HTML giusto, senza dover inserire domande o risposte.

• /directlogin • /directregister • /directpassreset • /dashboard

Si occupano del reindirizzamento dell'utente tra le diverse pagine web HTML in risposta a determinati input o azioni.

Dashboard.js: È il modulo che contiene il javascript della dashboard, la quale è definita in HTML come una pagina contentente tre tabs, rispettivamente contenenti una pagina per porre e visualizzare le domande, una pagina per rispondere alla domanda mostrata a schermo e una pagina per visualizzare la classifica. All'interno di questo modulo, sono definite le seguenti funzioni con le rispettive responsabilità:

- **loadAllTabContents(), initializeTabForms():** Funzioni che si occupano di inizializzare i contenuti di tutti i tab e renderli accessibili, per poi inserire degli ActionListener su ogni form dei vari tab (tra i vari form si trovano quelli che si occupano dell'invio di domande, risposte o refressh delle pagine).

- **handleFormSubmission(event, method):** Funzione che viene chiamata dagli ActionListener dei form. Si occupa di fermare e intercettare la richiesta, in modo da eseguirla poi in modo controllato e collocare la pagina

html ricevuta nel tab corretto, evitando difetti di visualizzazione nella pagina.

- **loadValidate(questionId, checked), sendBest(questionId, answerId, id_order_json), sendHuman(humanId, questionId):** Funzioni che si occupano di gestire visivamente le scelte sulle risposte da parte dell'utente, e inviare correttamente tali scelte al frontend

- **checkNewAnswers(), refreshLeaderboard():** Funzioni che si occupano rispettivamente di aggiornare le notifiche all'utente quando una domanda ha ricevuto tutte le risposte e di aggiornare la classifica. Entrambe le funzioni vengono chiamate ogni 5 secondi, in modo da mostrare notifiche e classifica in tempo reale.

Ognuna di queste funzioni incorpora un controllo per verificare se il token di sessione sia scaduto o assente, per interrompere l'operazione e dirigere l'utente di nuovo alla pagina di login. Inoltre incorporano un controllo per fare in modo che alla chiusura della pagina venga effettuata una richiesta all'endpoint /logout, in mdo che l'utente rilasci le risorse assegnate anche se non esegue un logout esplicito.

Profile.js: È il modulo che contiene il javascript del profilo, raggiungibile dalla pagina HTML della Dashboard, che contiene tutti i dati dell'utente (Username, punteggio, statistiche, friendcode). Questo modulo si occupa di gestire la logica del tasto che permette all'utente, alla pressione, di copiare il codice amico direttamente nella clipboard della propria pagina, per poterlo incollare altrove.

5 Sviluppi futuri

Future espansioni del progetto comprendono:

- **IA personalizzate:** Nella versione attuale, le operazioni IA vengono eseguite da Ollama con modello gemma3:1b. Come passo successivo, è possibile integrare le tre funzionalità dell'IA (controllo pertinenza tra domanda e tema, produzione di una risposta data una domanda e umanizzazione di una risposta per confondere le risposte IA con quelle di altri utenti) con delle intelligenze artificiali distinte e specializzate sullo specifico compito, rendendo le operazioni più efficenti.

- **Adattamento al mobile:** I templates HTML sono attualmente ideati per una visualizzazione da Desktop. Il passo successivo è adattare l'interfaccia anche ai dispositivi mobili.

- **Temi personalizzabili:** Attualmente, i temi delle domande sono prefissati, e l'utente può scegliere quale tra i 17 temi disponibili sia il più adatto alla domanda che vuole porre. Uno sviluppo futuro può consistere nel rendere il tema una scelta più libera per l'utente, potenzialmente inserendo una valutazione da un'altra IA specializzata che possa valutare la validità del tema come affine alla cultura italiana.

- **Assegnazione punti codice amico:** Al momento, alla registrazione di un nuovo utente con il friendcode, unicamente il nuovo utente guadagnerà punti. Per incentivare più utenti ad usare questo codice, sarebbe possibile conferire all'utente, proprietario del codice, dei punti nel momento in cui il nuovo utente abbia posto una domanda o le abbia risposto.

- **Tutorial:** Attualmente, le istruzioni su come usare l'applicazione sono situate sulla pagina principale. Per una futura versione si può inserire, alla creazione dell'utente, un breve tutorial interattivo che mostri le differenti possibilità.

- **Personalizzazione profilo:** Attualmente, la pagina del profilo contiene unicamente i dati dell'utente e la possibilità di cambiare la propria passsword. Sarebbe posibile, in futuro, inserire delle personalizzazioni aggiuntive, come impostare la propria foto profilo (attualmente predefinita e inalterabile) o selezionare un tema chiaro/scuro per la pagina.