Progetto Esame Ingegneria Informatica

Gabriel Rodriguez 1984875 and **Andrea Ravanetti 2002857**

1 Obiettivo del progetto

Il progetto consiste nello sviluppo di un programma in grado di estrarre informazioni da un database utilizzando un modello di linguaggio di grandi dimensioni (LLM).

2 Database

Il database è composto da quattro tabelle:

- **directors**: contiene i dati dei registi, con i campi `director` (nome del regista, chiave primaria) e `age` (età del regista).
- **movies**: composta dai campi `titolo` (titolo del film, chiave primaria), `anno` (anno di uscita), `director` (nome del regista, chiave esterna verso `directors`) e `genre` (genere del film). Sono presenti inoltre due vincoli di unicità, `UC_film` e `UC_film2`, che garantiscono l'unicità del film. Si noti che `titolo` e `anno` sono gli unici termini in italiano, per una specifica richiesta del tester.
- **platform**: contiene la piattaforma che ospita i film, con il campo `platform_name` come chiave primaria.
- **relation_platform_film**: tabella di relazione con i campi `id_relation` (chiave primaria autoincrementale), `titolo` (chiave esterna verso `movies`), `platform1` e `platform2` (chiavi esterne verso `platform`). È presente un ulteriore vincolo di unicità sul nome del film.

3 Organizzazione del codice

3.1 Backend

All'interno della cartella `backend/src/backend` sono presenti diversi file fondamentali per la gestione delle funzionalità del backend:

- `add_request.py`: gestisce il file `data.tsv`, contenente i dati iniziali per il popolamento del database. La funzione principale è `add_from_row`, che si occupa dell'inserimento nelle tabelle tramite le funzioni `add_entry_to_movies`, `add_entry_to_directors` e `add_entry_to_platforms`.
- `ai_support.py`: contiene funzioni di supporto per l'integrazione con il LLM. In particolare, le funzioni `SQLinator` e `clean_ai` permettono di ripulire l'output del modello per ottenere query SQL chiare e leggibili. La funzione `check_validation` verifica la validità delle richieste.
- `backend.py`: gestisce le richieste provenienti dal frontend. Tra le funzioni principali troviamo:
  - `init_tables`: inizializza il popolamento delle tabelle e avvia il modello Ollama.
  - `read_root`: verifica la connessione iniziale al sito.
  - `health_check`: verifica lo stato del sistema.
  - `get_schema_summary`: fornisce informazioni sul database.
  - `search`: riceve una domanda in linguaggio naturale, la passa al LLM, pulisce l'output SQL, ne verifica la validità, esegue la query e restituisce il risultato al frontend.
  - `sql_search`: esegue direttamente query SQL ricevute dal frontend dopo averne verificato la validità.
  - `add`: riceve dati in input, ne verifica la completezza e aggiorna il database.

- `get_request.py`: contiene la funzione `schema_summary`, che esplicita la struttura del database per il frontend.
- `init_tables.py`: contiene la funzione omonima che estrae informazioni da `data.tsv` per popolare il database.
- `mariadb_manager.py`: gestisce la connessione con MariaDB. Le funzioni principali sono:
  - `connect`: stabilisce la connessione al database.
  - `execute_query`: esegue query SQL.
  - `execute_insert`: esegue operazioni di inserimento.
- `schema_info.py`: contiene tre funzioni fondamentali:
  - `get_db_struct`: recupera informazioni sul database tramite chiamate a `information_schema` e crea una struttura dati a dizionario.
  - `ai_database_string`: prepara le informazioni per il LLM.
  - `stringify_structure`: descrive in linguaggio naturale la struttura del database per il LLM.
- `school.py`: definisce le classi e i builder necessari per la gestione delle strutture complesse nel backend.

3.2 Frontend

Nella cartella `frontend` è presente il file `frontend.py`, che gestisce l'interazione con l'utente e invia le richieste al backend. Le funzioni principali sono:

- `get_response`: ottiene il summary del database.
- `landing_page`: carica la pagina iniziale del sito.
- `schema_summary`: invia la richiesta di summary al backend.
- `search` e `sql_search`: effettuano controlli preliminari sugli input e inviano le richieste appropriate al backend.
- `add`: verifica i dati in input e invia la richiesta di inserimento al backend.

Sono inoltre presenti i template HTML che mostrano i risultati in pagine dedicate.

3.3 Setup e orchestrazione

Il progetto include vari file di configurazione per l'ambiente, tra cui `Dockerfile` e `requirements.txt`, orchestrati tramite `docker-compose.yaml`.

Il processo di avvio prevede:

1. Download delle immagini Docker per Ollama e MariaDB.
2. Avvio dei container con healthcheck per verificarne lo stato.
3. Avvio del backend.
4. Avvio del frontend.