Progetto di Text2SQL – Laboratorio di Ingegneria Informatica: Interrogazione di Database tramite Linguaggio Naturale

Emanuele Cocchi e Alessandro Corvi
Sapienza Università di Roma
cocchi.1967642@studenti.uniroma1.it
corvi.1946241@studenti.uniroma1.it

1 Obiettivo del Progetto

L'obiettivo del progetto è lo sviluppo di un'applicazione web in grado di permettere agli utenti di interrogare un database relazionale utilizzando il linguaggio naturale. L'applicazione si occupa di tradurre automaticamente interrogazioni in italiano in query SQL, eseguire tali query su un database e restituire i risultati in modo leggibile. Nel progetto, la conversione della domanda in italiano alla query SQL viene eseguita da un LLM (gemma3:1b-it-qat).

2 Organizzazione del Database

Il database **movie_catalog**, all'interno del DBMS MariaDB, è strutturato attorno a tre tabelle principali:

- **directors** (id, nome, eta): contiene le informazioni che riguardano il singolo regista, identificato univocamenemte da id (primary key) e anche da nome (unique).
- **platforms** (id, nome): contiene le informazioni che riguardano la singola piattaforma streaming, identificata univocamente da id (primary key) e anche da nome (unique).
- **movies** (id, titolo, anno, genere, id_director, id_platform1, id_platform2): contiene le informazioni che riguardano il singolo film, identificato univocamente da id (primary key) e anche da titolo (unique).

Sono presenti tre foreign key all'interno della tabella **movies**: id_director si riferisce al campo id della tabella **directors**, id_platform1 e id_platform2 si riferiscono al campo id della tabella **platforms** (un film può essere su al più due piattaforme) per garantire l'integrità referenziale.

3 Organizzazione del codice

Il progetto è suddiviso in cinque componenti principali: `backend`, `frontend` e `mariadb_init`, `load_db`, `text_to_sql`, ciascuno organizzato in modo modulare e coerente con le rispettive responsabilità.

3.1 Backend

La cartella `backend` è strutturata come segue:

- `backend/src/backend/`:
  - `connection_manager.py`: definisce la classe `ConnectionManager`, responsabile della gestione della connessione al database MariaDB e dell'esecuzione delle query SQL.
  - `models.py`: contiene le classi di modello (derivate da `BaseModel` di Pydantic) utilizzate per definire le strutture dati delle API tra backend e frontend.
  - `backend.py`: punto di ingresso dell'applicazione FastAPI. Gestisce gli endpoint REST, interagisce con il database tramite `ConnectionManager`, interagisce con il modello di IA tramite `ModelController` (definito in `text_to_sql/model_controller.py`) e comunica con il frontend.

3.2 Frontend

La cartella `frontend` è organizzata come segue:

- `frontend/src/frontend/`:
  - `frontend.py`: gestisce gli endpoint web e comunica con il backend usando la libreria `requests`. Interagisce con le pagine HTML tramite `Jinja2Templates`.

- `frontend/templates/`:
  - `index.html`: pagina principale in cui l'utente può inserire una domanda in linguaggio naturale e accedere alle altre funzionalità.
  - `schema_summary.html`: visualizza lo schema del database.
  - `search.html`: mostra i risultati ottenuti dall'esecuzione della query SQL corrispondente alla domanda.
  - `add.html`: consente l'inserimento di nuovi dati nel database.
  - `sql_search.html`: consente l'inserimento di una query scritta in SQL e mostra i risultati corrispondenti.

3.3 Mariadb_init

La cartella `mariadb_init` contiene lo script di inizializzazione del database:

- `init.sql`: definisce la struttura delle tabelle per il database all'interno di MariaDB. Lo script viene eseguito automaticamente all'avvio del container per predisporre il database.

3.4 Load_db

La cartella `load_db` contiene lo script che serve per caricare i dati, estratti da un file, all'interno del database:

- `data.tsv`: file che contiene i dati da caricare all'interno del database.
- `load_db.py`: estrae i dati dal file `data.tsv` e li inserisce nelle giuste tabelle del database .

3.5 Text_to_sql

La cartella `text_to_sql` contiene lo script che si occupa della conversione da testo a SQL:

- `text_to_sql/src/`:
  - `model_controller.py`: definisce la classe `ModelController`, responsabile della gestione e della comunicazione con il modello di IA del framework Ollama.