Quaderno:
Sviluppo di un'applicazione web con Streamlit e MySQL

Obiettivo

Creare un'applicazione web in Python (Streamlit) in grado di interagire con un database MySQL in modo da eseguire interrogazioni in base alle interazioni dell'utente.

Fase preliminare

- Avviare MySQL tramite Docker o XAMPP (vedi materiale *Live Coding* e *Laboratorio 6*)
- Importare il database
- Creare e avviare il progetto Streamlit (nuovo progetto o dal branch *base* del repository)
- Verificare la connessione al db attraverso le credenziali
- ***Suggerimento***: usare *st.write()* per investigare i risultati ottenuti

Descrizione del database

La base di dati si chiama PALESTRA e riguarda le attività di una palestra. Essa è caratterizzata dal seguente schema logico (le chiavi primarie sono sottolineate):

ISTRUTTORE (CodFisc, Nome, Cognome, DataNascita, Email, Telefono*)

CORSI (CodC, Nome, Tipo, Livello)

PROGRAMMA (CodFisc, Giorno, OraInizio, Durata, CodC, Sala)

Utilizzare l'interfaccia di phpMyAdmin per importare lo script e verificare che sia andato a buon fine. Usare l'interfaccia per esplorare la base di dati, le tabelle disponibili e i dati salvati.

Esercizi

Creare un'applicazione multi-pagina per visualizzare le principali informazioni contenute nel database e permettere all'utente di aggiungerne di nuove. In particolare:

1. Creazione di un *Homepage* personalizzata che sfrutti la sintassi base del markdown (o gli elementi Streamlit) per introdurre il laboratorio, l'obiettivo e lo studente.

2. Creazione di una pagina per la visualizzazione e filtraggio dei corsi disponibili. La pagina deve essere supportata da due *metric* per mostrare il numero di corsi e di tipi distinti disponibili. I widget di input devono essere creati in modo da proporre come opzioni le informazioni contenute già a database. L'utente deve poter visualizzare le informazioni sui corsi filtrando per più categorie (i.e., *Tipo*) e deve poter specificare il range di livello a cui è interessato. In un expander separato, visualizzare i programmi delle lezioni per i corsi selezionati. In caso di risultati vuoti, bisogna stampare un errore/warning associato.

3. Creazione di una pagina per la visualizzazione degli istruttori disponibili. L'utente deve avere la possibilità di filtrare digitando il cognome dell'istruttore e utilizzando un date range per scegliere in base alla data di nascita (***hint:*** usare datetime.date() per impostare il date_input e passare la data come stringa nell'interrogazione). La visualizzazione non deve essere una tabella complessiva, ma divisa elemento per elemento (creare un dataframe e usare *df.iterrows per stampare una row alla volta, vedi Lab 6*). In caso di risultati vuoti, bisogna visualizzare un messaggio associato.

4. Creazione di una pagina per visualizzare grafici riguardanti le lezioni programmate. La pagina deve contenere due grafici: un Bar Chart che riporti il numero di lezioni per ogni slot di tempo e un Line Chart che riporti il numero di lezioni programmate in base al giorno della settimana.

5. Creazione di una pagina per l'inserimento di nuovi corsi attraverso un form adatto. Usare un form d'inserimento che richiede tutti i dati necessari all'inserimento di un nuovo corso nella base di dati (CodC, Nome, Tipo, Livello). L'applicazione deve verificare che tutti i campi siano valorizzati e che il valore dell'attributo Livello sia un numero intero compreso tra 1 e 4 (***hint:*** usare *number_input*). In caso di dati mancanti, chiave duplicata o altri errori, l'applicazione deve generare un messaggio d'errore. Se invece i dati inseriti sono corretti e l'operazione d'inserimento va a buon fine, si deve visualizzare un messaggio di corretto inserimento.

6. Creazione di un form per l'inserimento di una nuova lezione settimanale nella tabella PROGRAMMA. Il form deve permettere di inserire tutti i campi necessari (CodFisc, Giorno, OrarioInizio, Durata, CodC, Sala) relativi alla programmazione di una nuova lezione. La selezione dell'istruttore deve avvenire tramite un menù a tendina contenente il codice fiscale dei possibili istruttori generato dal contenuto della tabella della base di dati. In modo analogo, anche la selezione del corso deve avvenire tramite un menù a tendina popolato dalla base di dati. Gli altri campi sono invece campi popolati manualmente dall'utente, utilizzando i widget più adatti (e.g., slider per OraInizio e Durata) o quelli testuali. L'applicazione delle verificare che l'utente non cerchi di inserire nel programma lezioni che durino più di 60 minuti e che il giorno indicato sia un giorno compreso tra Lunedì e Venerdì. L'inserimento di una nuova lezione in programma deve essere consentito ed eseguito se e solo se non sono in programma altre lezioni per lo stesso corso nello stesso giorno della settimana (***hint:*** utilizzare i valori di input per effettuare l'interrogazione e verificare che non ci siano record). Se la richiesta di inserimento rispetta i vincoli indicati e l'inserimento termina correttamente, si deve visualizzare un messaggio di corretto inserimento, altrimenti si deve notificare un messaggio d'errore (il messaggio d'errore deve riportare il tipo di problema che ha comportato l'errore).

Tutte le pagine devono essere personalizzate con elementi di testo (utilizzando markdown o i widget pre-impostati di Streamlit) in modo da avere titoli, sottotitoli e paragrafi che evidenzino cosa viene rappresentato. Oltre a generare le interrogazioni corrette, per rendere la visione e l'interfaccia più intuitiva e organizzata, devono essere utilizzati i principali elementi di layout: expander, colonne, tab.