Homework no. 4: Development of a web application with Streamlit and MySQL

Objective

Create a web application in Python (Streamlit) capable of interacting with a MySQL database in order to execute queries based on user interactions.

Start MySQL via Docker or XAMPP (see Live Coding material and Lab 5)

Preliminary phase

- Start MySQL via Docker or XAMPP (see Live Coding material and Lab 5)
- Import the database
- Create and start the Streamlit project (new project or from the base branch of the repository)
- Verify the connection to the database with the credentials
- ***Suggestion***: use *st.write()* to investigate the results obtained

Database description

The database is called GYM and concerns the activities of a gym. It is characterized by the following logical scheme (the primary keys are underlined):

INSTRUCTOR (<u>FisCode</u>, Name, Surname, BirthDate, Email, Telephone*)

COURSES (<u>CodC</u>, Name, CType, Level)

PROGRAM (<u>FisCode</u>, <u>Day</u>, <u>StartTime</u>, Duration, CodC, Room)

Use the phpMyAdmin interface to import the script and verify that it was successful. Use the interface to explore the database, available tables and saved data.

Exercises

Create a multi-page application to display the main information contained in the database and allow the user to add new information. In particular:

1. Creation of a customized *Homepage* that uses the basic markdown syntax (or Streamlit elements) to introduce the laboratory/notebook, the objective and the student. Below, report two graphs regarding the scheduled lessons: an **Area Chart** showing the number of lessons for each time slot and a **Bar Chart** showing the number of scheduled lessons based on the day of the week.

2. Creation of a page for viewing and filtering available courses. The page must be accompanied by two *metrics* to show the number of courses and distinct types available. The input widgets must be implemented to provide as selectable options the information already present in the database. The user must be able to view information on courses by filtering by multiple categories (i.e., C*Type*) and must be able to specify the level range in which she/he is interested. In a separate expander, view lesson plans for the selected courses and the corresponding instructor's first and last name (in the same column) and email. In case of empty results, an error/warning must be printed.

3. Create a page to display available instructors. The user must be able to filter by typing the instructor's last name and using a date range to choose based on the date of birth (**hint:** use *datetime.date()* to set the date_input and pass the date as a string in the query). The view should not be an aggregate table, but divided element by element (create a dataframe and use *df.iterrows to print one row at a time, see Lab 5*). Add an icon for each result. In case of empty results, an error message/warning must be displayed.

4. Creation of a page to insert new courses through a suitable form. Use an entry form that requests all the data necessary for inserting a new course into the database (CodC, Name, CType, Level). The application must verify that all fields are filled in and that the Level is an integer between 1 and 4 (**hint:** use *number_input)*. CodC must follow the correct format (i.e., start with "CT"). In case of missing data, duplicate key or other errors, the application should generate an error message. However, if the data entered is correct and the insertion operation is successful, a correct insertion message should be displayed.

5. Creation of a form for inserting a new weekly lesson in the PROGRAM table. The form must allow you to enter all the necessary fields (FisCode, Day, StartTime, Duration, CodC, Room) relating to the programming of a new lesson. The selection of the instructor must be done via a drop-down menu containing the fiscal code of the possible instructors generated from the contents of the database table. Similarly, the selection of the course must also take place via a drop-down menu populated from the database. The other fields are populated manually by the user, using the most suitable widgets (e.g., sliders for StartTime and Duration) or textual ones. The application verifies that the user does not try to insert lessons in the program that last more than 60 minutes and that the day indicated is a day between Monday and Friday. The insertion of a new scheduled lesson must be allowed and executed if and only if no other lessons are scheduled for the same course on the same day of the week (**hint:** use the input values to carry out the query and verify that there are no records). If the insertion request respects the constraints and the insert operation is executed correctly, a success message must be displayed, otherwise an error message must be notified (the error message must indicate the type of problem that led to the failure).

All pages must be customized with text elements (using markdown or Streamlit pre-made widgets) in order to have titles, subtitles and paragraphs that highlight what is being represented. In addition to generating the correct queries, to make the interface more intuitive and organized, the main layout elements must be used: expanders, columns, tabs.