Database Interface - CRUD Operations with Flask for Python and MySQL

Manta Mădălin-Ștefan, 432Fa  
Faculty of Electronics, Telecommunications and Information Technology

**This document presents the Music Management Application, a web-based system designed to provide a simple and intuitive interface for interacting with a relational database. The application implements CRUD (Create, Read, Update, Delete) operations for managing musical data, including albums, musicians, and their relationships. Developed using Python Flask for the backend, HTML, CSS for the frontend, and MySQL as the database, the project focuses on usability and functionality. The interface enables users to seamlessly interact with the database through a user-friendly web-based graphical interface. This report details the database structure, application design, and implementation of core functionalities, highlighting the effectiveness of the approach to database management.**

1. Introduction

The Music Management Application was developed to provide an intuitive and effective web-based interface for managing musical data stored in a relational database. The application allows users to interact with the database through CRUD operations on entities such as albums, musicians, and their relationships.

Built with Python Flask for the backend, HTML/CSS for the frontend, and MySQL as the database, the project focuses on demonstrating the fundamentals of web application development and database management. The application is structured to ensure ease of use, making it accessible for both beginners and advanced users looking to explore database integration in web applications. Additionally, the interface prioritizes clarity and usability, ensuring a seamless user experience by allowing users to manage musical data efficiently.

1. DATABASE STRUCTURE

The relational database is implemented in MySQL and consists of the following tables:

Albums: Stores details about albums, including their titles and release years.

Musicians: Contains information about musicians, including their names and music genres.

Albums\_Musicians: A junction table representing the many-to-many relationship between albums and musicians.

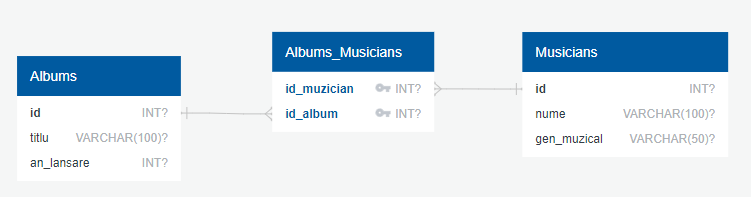


Fig. 1. Database diagram

1. Implementation

The application architecture follows the Model-View-Controller (MVC) pattern:

* Model (Database): MySQL tables store the relational data.
* View (Frontend): HTML templates with CSS styles provide the UI.
* Controller (Flask Backend): Python routes handle requests and database operations.
  1. *Functionalities*

The Music Management Application provides an intuitive web-based interface for managing musical data. Users can perform the following operations:

**1. Musicians Management**

* View the list of musicians.
* Add new musicians.
* Edit musician details.
* Delete musicians, ensuring database integrity.

**2. Albums Management**

* View the list of albums.
* Add new albums.
* Edit album details.
* Delete albums, ensuring database integrity.

**3. Relationships Management**

* View the relationships between musicians and albums.
* Add a relationship between a musician and an album.
* Delete relationships.
  1. *Code Structure*

1. Database Connection: The get\_db\_connection() function establishes a connection to MySQL

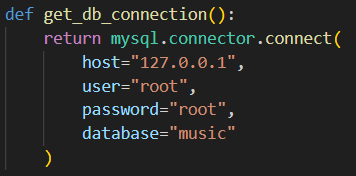


Fig 2. Database Connection

1. CRUD operations Each operation is handled via Flask routes, allowing communication between the frontend (HTML) and the MySQL database.

* Adding a musician involves inserting data into the Musicians table:

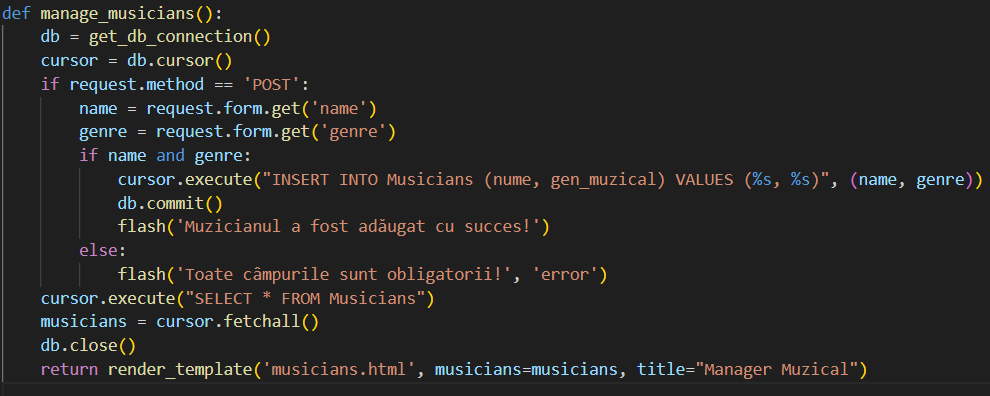


Fig. 3. Adding a musician

* Modifying a musician updates the data in the database:

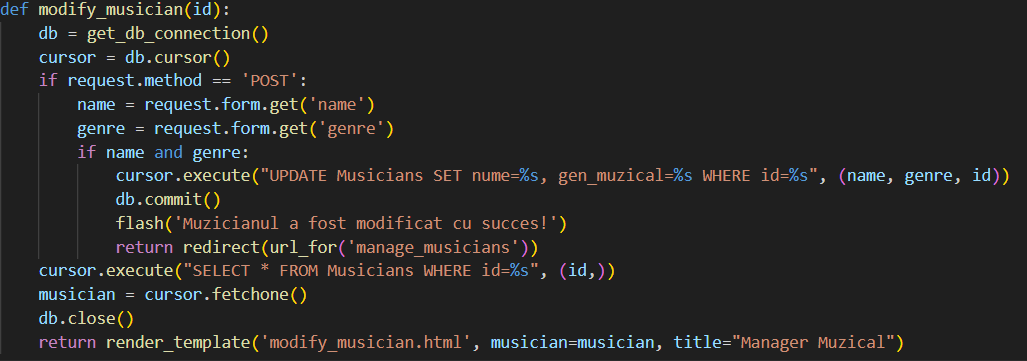


Fig. 4 Modifying a musician

* Deleting a musician ensures referential integrity by removing linked records from Albums\_Musicians before deletion:

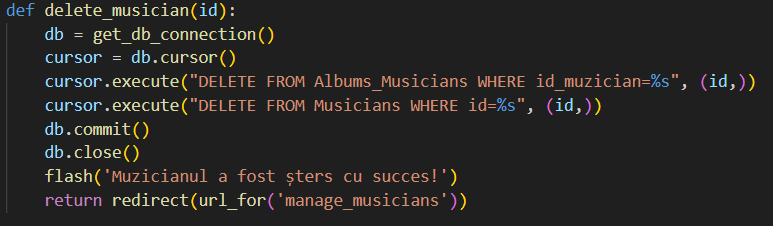


Fig. 5. Deleting a musician

* 1. *User Interface*

The frontend interface is built using HTML templates and CSS, providing:

1. Tables to display data from MySQL.
2. Forms for user input..
3. Buttons for triggering CRUD operations.
4. Flask Flash Messages for user feedback
5. Results

The application was tested for all CRUD operations, ensuring the correctness of database interactions and the integrity of data. Key test cases include:

**1. Adding a Musician**

* **Input:** Name = "Eminem", Genre = "Rap".
* **Steps:** Enter the musician's name and genre into the input fields and click "Add Musician"".
* **Result:** The new musician appears in the list and is stored in the database.

**2. Deleting an Album**

* **Input:** ID = 3 (Album "Revival").
* **Steps:** Select the album and click "Delete" from its right side.
* **Result:** The album is removed from the list and the database.

**3. Creating a Relationship**

* **Input:** Musician = "Eminem", Album = "The Marshall Mathers LP".
* **Steps:** Select the musician and album, then click "Create Relationship".
* **Result:** The relationship is added to the Albums\_Musicians table.

**4. Modifying a Musician**

* **Input:** Name = "El Nino", Genre = "Hip Hop"
* **Steps:** Select the wanted artist, enter the new details in the TextBox and click "Modify Musician".
* **Result:** Our desired musician has now the changed attributes.

All test cases were executed successfully, and the application performed as expected, maintaining database integrity and providing a seamless user experience.



Fig. 6. Main interface

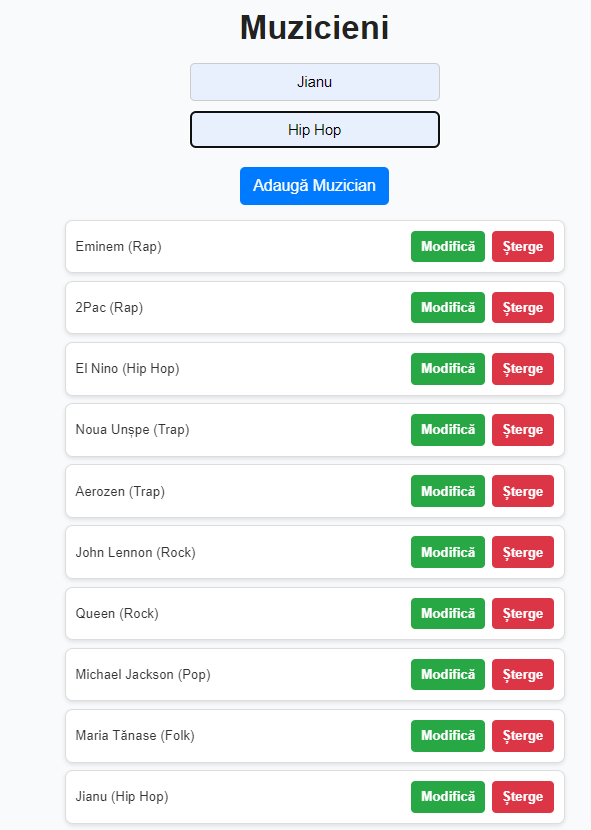


Fig. 4. Adding a Musician

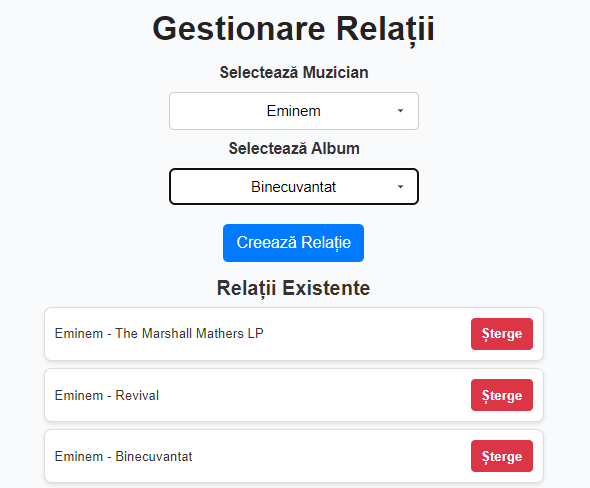


Fig. 5. Creating a Relationship



Fig. 6. Modifying a Musician

1. Conclusion

The Music Management Application successfully demonstrates CRUD operations on a MySQL database using a Flask web application. The interface is user-friendly and provides essential functionalities such as adding, viewing, editing, and deleting data. In the future, we could probably see some enhancements as:

* Advanced search and filtering capabilities.
* Improved UI styling with Bootstrap or React.
* Additional features like export and reporting tools.

References

[Fig. 1] <https://app.quickdatabasediagrams.com/#/>

MySQL Documentation, "MySQL 8.0 Reference Manual," 2023. [Online]. Available: <https://dev.mysql.com/doc/>.

Flask Documentation, "Flask Framework Guide," 2023. [Online]. Available: <https://flask.palletsprojects.com/>.

Python MySQL Connector, "MySQL Connector for Python," 2023. [Online]. Available: <https://dev.mysql.com/downloads/connector/python/>.