BookOrg

Short-term database interaction project for the subject of PV

Michal Bielina

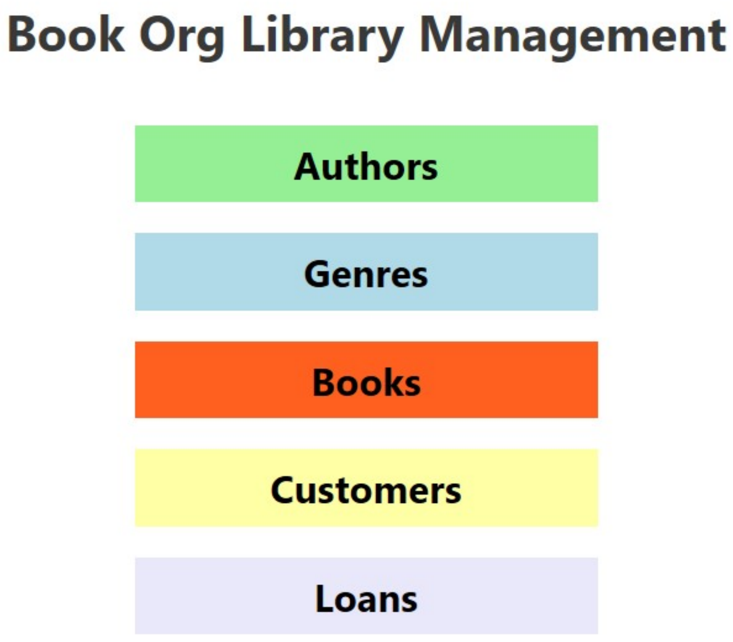
C4a
Informační technologie - Programování a digitální technologie
(Information Technology - Programming and Digital Technologies)

Střední průmyslová škola elektrotechnická
Praha 2, Ječná 30
2026

Table of Contents

1. Introduction

This documentation describes the development and features of the **BookOrg** project — a lightweight library management system designed to simplify the organization of books, authors, genres, and customer records. The application allows users to manage the entire lifecycle of a book loan, from automated stock deduction to tracking returned items. Developed using the WPF (Windows Presentation Foundation) framework and .NET 8.0, the project emphasizes data integrity, persistence via a Microsoft SQL Server backend, and a clean, modular code architecture.



Picture 1: Main Menu

2. User specifications and requirements

The system is designed to provide a centralized interface for library administrators to perform the following tasks:

- Resource Management: Create, read, update, and delete (CRUD) records for Books, Authors, Genres, and Customers.
- Automated Loan Processing: Manage the lifecycle of book loans, including automatic stock deduction upon loan creation and restoration upon return.
- Bulk Data Integration: Import large datasets of Authors, Genres, or Customers from external CSV files.
- Relational Integrity: Maintain many-to-many relationships between books and their respective authors and genres.

3. Application architecture

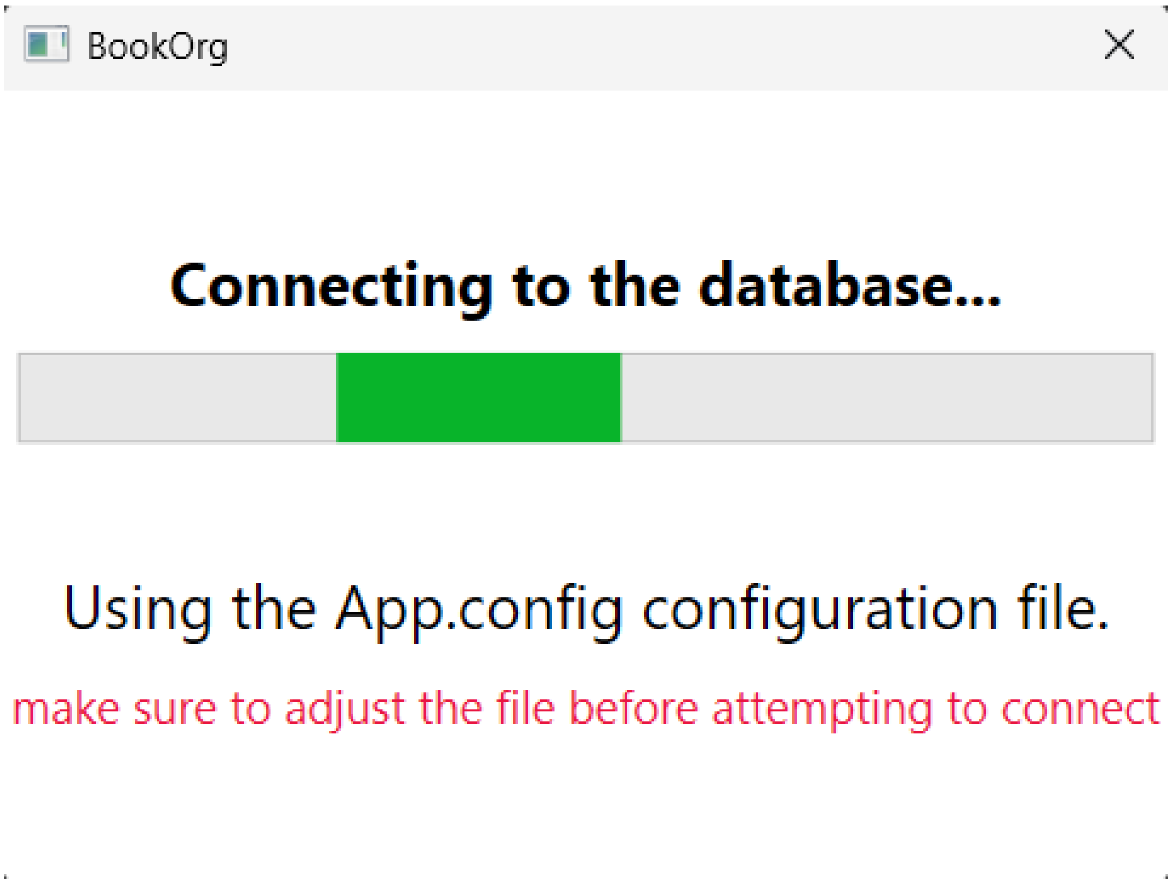
BookOrg uses a modular architecture to ensure clean separation of concerns:
Presentation Layer (WPF): XAML for UI definitions and Code-Behind with INotifyPropertyChanged to handle data binding and UI updates.

- Logic Layer (DAO Pattern): The Data Access Object (DAO) pattern is used to abstract database operations. Each entity has a dedicated DAO (e.g., BookDAO, LoanDAO) that inherits from DAOBase<T>.
- Connection Management: A Factory pattern (SqlServerConnectionFactory) manages the lifecycle of SQL connections based on App.config settings.
- Safety Layer: A static SafeExecutor utility wraps critical operations in try-catch-finally blocks to provide standardized error reporting and logging.

4. Behavioral logic

The application follows a strict operational flow:

1. Initialization State: The ConnectionWindow attempts to open a database connection using App.config.
2. Navigation State: Upon success, the MainWindow hosts a MainMenuControl for module selection.
3. Loan Lifecycle State:
   - New Loan: Decrements books_in_stock in the book table.
   - Return Loan: Updates status to "returned," sets loan_returned_date, and increments books_in_stock.
   - Deletion: Deleting an active loan restores the book stock.


BookOrg

Connecting to the database...

Using the App.config configuration file.

make sure to adjust the file before attempting to connect

Picture 2: Connecting Window

5. Database system description

The BookOrg database is implemented in Microsoft SQL Server. Major operations are encapsulated in transactions to ensure that multi-table updates are processed atomically.

Database Tables

- genre: Stores literary categories. Attributes: id (Primary Key) and genre_name.
- author: Stores unique author identities. Attributes: id (Primary Key) and author_name.
- book: Manages primary book data and inventory. Attributes: id (Primary Key), title, is_available (availability flag), price, and books_in_stock.
- contribution: A junction table for the many-to-many relationship between books and authors. Attributes: id, book_id (Foreign Key), and author_id (Foreign Key).
- classification: A junction table facilitating the many-to-many relationship between books and genres. Attributes: id, book_id (Foreign Key), and genre_id (Foreign Key).
- customer: Stores information about library members. Attributes: id (Primary Key), first_name, last_name, and email.
- loan: Tracks borrowing transactions. Attributes: id (Primary Key), book_id (FK), customer_id (FK), total_loan_price, loan_status (active/overdue/returned), loan_start_date, loan_due_date, and loan_returned_date.

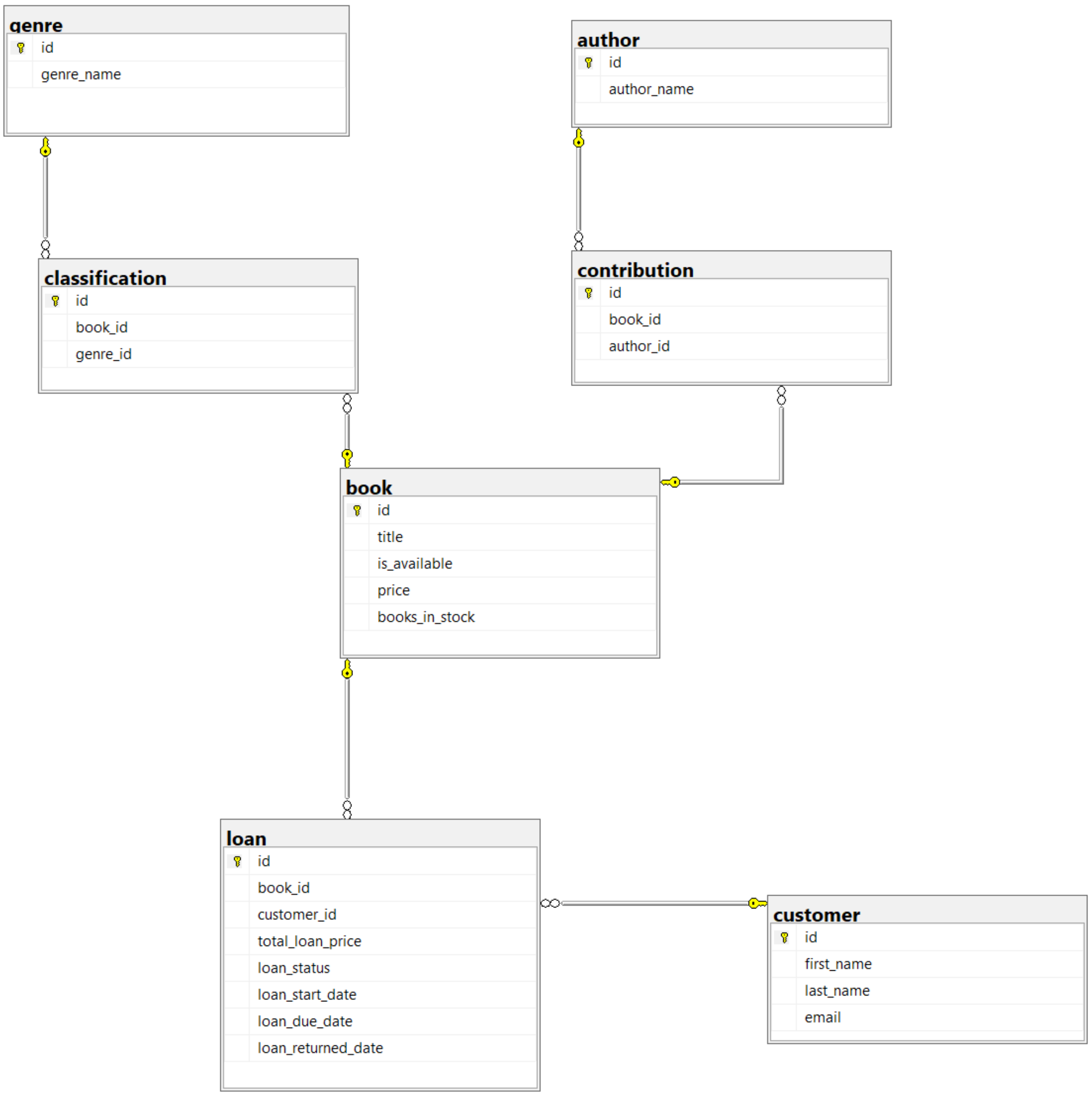
Database Views

The system utilizes views to simplify data retrieval for the user interface by aggregating related information:

- view_books: Aggregates book records with their associated authors and genres into comma-separated strings for high-level overviews.
- view_ongoing_loans: Joins loan, book, and customer data for all entries with a status of "active" or "overdue." It is used by the LoanDAO to populate the active loans list in the UI.
- view_returned_loans: Joins the same tables but filters for entries with a "returned" status. It is used to display the historical record of completed transactions in the application.

Additional Information

One can observe that the loan only allows for one book for one customer at the time. This might seem strange since the customer could theoretically loan multiple books at ones. While this is true, the single-book principle is actually a wanted behaviour. Because each loan has a defined start and due date, this approach actually allows for each book to be returned at different times.



Picture 3: Database diagram

6. Csv data import specification

The BookOrg system allows users to fill the library database efficiently using external CSV files. This process is managed by the CsvImporter class, which interacts with entity-specific Data Access Objects (DAOs) to validate and persist the data.

Supported File Formats

The system currently supports importing data for three core entities. Each file must include a header row, as the importer is programmed to skip the first line of any source file.

- Authors (authors.csv):
    - Expected Columns: 1.
    - Required Header: Author Name.
    - Format: Each subsequent row should contain a single string representing the author's name (e.g., "J.K. Rowling").
- Genres (genres.csv):
    - Expected Columns: 1.
    - Required Header: Genre Name.
    - Format: Each row contains a single genre category (e.g., "Science Fiction", "Mystery").
- Customers (customers.csv):
    - Expected Columns: 3.
    - Required Headers: FirstName, LastName, Email.
    - Format: Rows must provide all three values separated by commas (e.g., "John,Doe,john.doe@example.com").

Technical Validation and Logic

- Strict Column Count: The CsvImporter verifies the number of columns in every row against the ColumnCount property defined in the target DAO. If a row does not match this count, a FormatException is thrown to ensure data integrity.
- Data Cleaning: During the import process, the system automatically trims leading and trailing whitespace from all string values before they are mapped to the database entities.

- Duplicate Prevention: When importing authors and genres, the underlying DAO logic checks for existing names in the database to prevent the creation of duplicate records.
- Encoding: All files are read using UTF-8 encoding to maintain support for special characters in names and titles.

7. Configuration and setup

Connection Parameters

Settings are managed in App.config under the <appSettings> section:

- DataSource: The IP or server name of the SQL instance.
- Database: The target database name.
- Login/Password: SQL Server authentication credentials.

Supplementary Setup Instructions

Beyond the README, ensure that:

1. File Permissions: The application has write access to its own directory to create the Logs/ folder.
2. SQL Permissions: The database user must have SELECT, INSERT, UPDATE, and DELETE permissions, as well as permission to execute VIEW queries.

More information about the project configuration can be found in the provided README file.

8. Error handling

The BookOrg application remains responsive and provides clear feedback even when database or runtime errors occur.

The SafeExecutor Architecture

The most important part of the safety system is the SafeExecutor static class. This utility wraps synchronous and asynchronous actions in a standardized try-catch-finally block:

- Catching: All exceptions are caught at the logic or UI boundary, preventing unhandled crashes.
- User Feedback: When an error is caught, the system triggers HandleError, which presents a MessageBox to the user containing a context-specific error message alongside the technical exception message.
- Cleanup: The finallyAction parameter allows for guaranteed execution of cleanup code. In this project, there is no usage for this part of the code but it was added so it can be used in the future or in different projects.

Thread-Safe Logging System

For debugging purposes, the application maintains a record of all events via the Logger class:

- Thread-Safety: Logging operations use a lockObject to prevent file access conflicts when multiple threads attempt to write simultaneously.
- Daily Log Rotation: Logs are automatically organized into a Logs/ directory, with files named by date.
- Detailed Entries: Each log entry includes a precise timestamp, a severity level (Info, Warning, or Error), and the specific message or stack trace provided by the catch block.

9. Third party libraries

The project maintains a minimal footprint by utilizing only essential libraries:

- Microsoft.Data.SqlClient: Enables the use of advanced features such as SqlTransaction, SqlDataReader, and asynchronous database commands.
- System.Configuration.ConfigurationManager: Used within the SqlServerConnectionFactory to securely retrieve sensitive connection strings and provider settings from the App.config file.
- WPF Toolkit / Standard Libraries: The application relies on the standard .NET 8.0 libraries for UI rendering (WPF), data binding via INotifyPropertyChanged, and collection management through ObservableCollection.

10. Project summary and additional information

Summary

BookOrg is a scalable foundation for library management. Using the DAO pattern and a central safety executor, the code remains maintainable and it is resistant to unhandled crashes. Future versions could expand on this architecture by adding user authentication and more complex reporting views.

Additional Information

This project was developed as part of a high-school project. It does not have any commercial ambitions and can be used as template or foundation for future development.

More information can be found on github. All pictures used, originate either from the project itself or the school website which is linked below.

Střední průmyslová škola elektrotechnická website: **https://www.spsejecna.cz/**
Github repository: **https://github.com/LupusLudit/BookOrg-D1**

Author: Michal Bielina

Contact: michal.bielina@centrum.cz