



Library 3.0

🕒 Date du sujet	@24 mars 2023 16:29
➤ Module	🏏 <u>4DOT</u>
▼ Type	Projet
⚙️ État	Prêt

Introduction

You are working as a developer for a municipal library that wants to modernize its book and borrower management system. Your mission is to develop a REST API that will efficiently manage the library's database, providing search, borrow and return functionality, as well as detailed borrower information.

You will also be responsible for developing the library's desktop application that allows the librarian to view previous information.

Subject

To do this, you will need to create a REST API with ASP .NET CORE and use a SQLite database that can only be accessed through the API. This database will be used to store information about books and borrowers.

You will also have to manage the current borrowing cases, allow the search of books and borrowers according to different criteria.

You will also have to create an application with WPF to allow librarians to have a visual application allowing them to find, modify, add or delete books, borrowers, etc...

This WPF application will communicate only with the API.

API REST (10 points)

The REST API must contain these methods:

Books:

1. Add a book
2. Modify an existing book
3. Delete an existing book
4. Select a book by its title
5. Select all books who contains a given word
6. Select all existing books in the database
7. Select all the books of the same author

Borrowers :

1. Add a borrower
2. Modify a borrower
3. Delete a borrower
4. Select a borrower by last name, first name or address
5. Select all existing borrowers in the database
6. Select all borrowers who contains a given word (can be used on first name and last name)
7. Select all borrowers who lives in a given adress (Like "Orléans")
8. Select all the borrowers of a book

Loan :

1. Borrow a book
2. Return a book (End of loan)
3. Modify a borrowed book
4. Delete a book loan
5. Get all books borrowed on a given date
6. Get all books borrowed between a start date and an end date
7. Get all books that are past their return date
8. Get all books that are loan by a borrower

AvaloniaUI Application (5 points)

The AvaloniaUI application allows the user to manage through a nice graphical interface everything that the API allows to realize.

Use buttons, texts, menus and sub-menus to make your GUI look nice.

For the visual part, remember to apply your UX course so that the navigation on your application is pleasant.

You are free to make your screens, as long as everything we ask is present.

All API functionality must be in the application

Home screen :

The welcome screen serves as an informative hub, highlighting returns scheduled for the day. It provides a quick overview of upcoming returns, ensuring you stay on top of your return management.

Additionally, the welcome screen offers easy navigation to other essential screens, such as borrowers, books, loans and returns, enabling seamless access to relevant information and streamlined workflow.

Book screen:

This screen provides a comprehensive display of all existing books, offering an efficient way to access and manage book information.

With this screen, you have the flexibility to delete, modify, or add new books, empowering you to effectively maintain and update your book inventory.

Borrowers screen:

This screen provides an overview of all existing borrowers, presenting a convenient way to access and manage user information.

With this screen, you can easily delete, modify, or add new users, granting you the ability to efficiently maintain and update user records.

Loan screen:

This screen offers a comprehensive view of all existing loans, providing a centralized hub for managing loan-related operations.

With this screen, you can effortlessly delete, modify, or add new loans, empowering you with complete control and flexibility over your loan management processes.

Return screen :

This screen provides functionality to validate the return of a book from a borrower, making the return process seamless.

Additionally, it offers a convenient view of current loans along with their respective return dates.

Notably, any loans that are overdue are automatically highlighted, ensuring their visibility and prioritization.

BONUS (5 points)

Feel free to improve the subject by implementing new features or improve the existing ones 🚀🐱🐶

Instructions

1. The application must be able to be compiled and executed correctly
2. The code must be organized in several files for each class
3. The database must be created when the application is launched if it does not already exist
4. The application must be able to manage the cases of ongoing borrowing, i.e. a book must not be deleted as long as it is borrowed by a borrower
5. The application must allow the search of books by title, author or ISBN
6. The application must allow the search of borrowers by last name, first name, address or email address.

Advice

1. Make sure your code is readable and well organized, using clear naming conventions and commenting on each method and class.
2. Use SQLite Studio to help you create your database. But remember that it must be created by the application if it does not exist!
3. Run test data through SQLite Studio to make sure your code works.
4. Make a group of 2, and share the work. One does the GUI, one does the API and coordinate!
5. Use GIT to make it easier to work in groups
6. For the different screens, a rather interesting technique, called "UserControl" can be found here: <https://docs.avaloniaui.net/docs/controls/usercontrol>
7. Think reuse, especially with UserControls, it will save you a lot of time!

Sample of models for your application

Below is an example of a table you can use for your application. Feel free to modify it according to your needs

Book :

- Attributes :
 - ISBN (Unique identifier)
 - Title
 - Author
 - Year of publication
 - Number of pages
 - Stock

Borrower :

- Attributes :
 - First Name
 - Last Name
 - Address (**Must be splitted on Number, Street, Zip code, Country and Land**)
 - Phone Number
 - Email
 - List of books borrowed (by himself)

Loan :

- Attributes :
 - Borrowed Date
 - Return Date
 - Borrowed Book
 - Borrower

Rules

This project must be your creation and done by your own. Plagiarism or copy / past are forbidden and will result of a 0.

This is project can be done in group of 2 students maximum.


Each group containing more than 2 members will be discarded and not evaluated.

Delivery

You must deliver a **ZIP** containing:

- Technical Documentation (.pdf)
- Projects files (.zip).

You must send your project to : artur.hoogers@ecole-it.com

 File extension should be strictly respected. If you deliver file in a different format than expected, the file will be entirely discarded.