Software Requirements Specification

for

Library

Prepared by Axel Caliari, Emmanuel Gay, Jiahong Wu and Victor Wang

Concordia University

23/05/2018

Table of Contents

1. Introduction 1

1.1 Purpose 1

1.2 Project Scope 1

1.3 Definitions, acronyms, and abbreviations 1

2. Overall Description 2

2.1 Product Perspective 2

2.2 Main functionalities 2

2.3 Operating Environment 4

3. System Features 4

3.1 Project Features 4

3.2 Use Case: make reservation 8

4. External Interface Requirements 10

4.1 User Interfaces 10

4.2 Hardware Interfaces 10

4.3 Software Interfaces 10

4.4 Communications Interfaces 10

# Introduction

## Purpose

The purpose of this document is to give a detailed description of the requirements for the “Library” software. It will illustrate the purpose and complete declaration for the development of the system. This document is primarily intended to be proposed to a project manager for its approval and a reference for developing the first version of the system for the development team.

Please do note that this project is realized by students, consequently people with higher expertise, experience and knowledge may find the application not plausible in a business or practical environment, not well organized and optimized, or incomplete. The purpose of this document is to showcase utilization of UML and design patterns that we, the students, have learnt and applied.

(Note to the professor: Please forgive us if the project seems to be more database-focused than object-focused.)

## Project Scope

The Library is a Java-based application designed for easy handling of librarian-specific tasks. With just a few clicks, the user can add, remove or search for a book in the library, browse the catalog, and make reservations for customers, making both the librarian’s and the borrower’s life easier. Items, customers and active reservations of book in the library are stored in a database and can be sorted in various ways by the user. The application is only accessible by registered librarians with the valid login credentials.

## Definitions, acronyms, and abbreviations

|  |  |
| --- | --- |
| User | Someone who interacts with the application |
| Admin | In this application, the admin is the user, and thus the librarian |
| Customer | Borrower, someone who borrows a book from the library |
| CRUD | Create, read, update and delete. User interface conventions facilitating viewing, searching and changing information |
| MVC | Model-view-control. Design/architectural pattern used for Library |
| GUI | Graphical User Interface |

# Overall Description

## Product Perspective

This system is a data-centric application. The main part, the Library application, will be used as a platform for interactions between the user and the database.

Desktop software

Database server

User

Library application

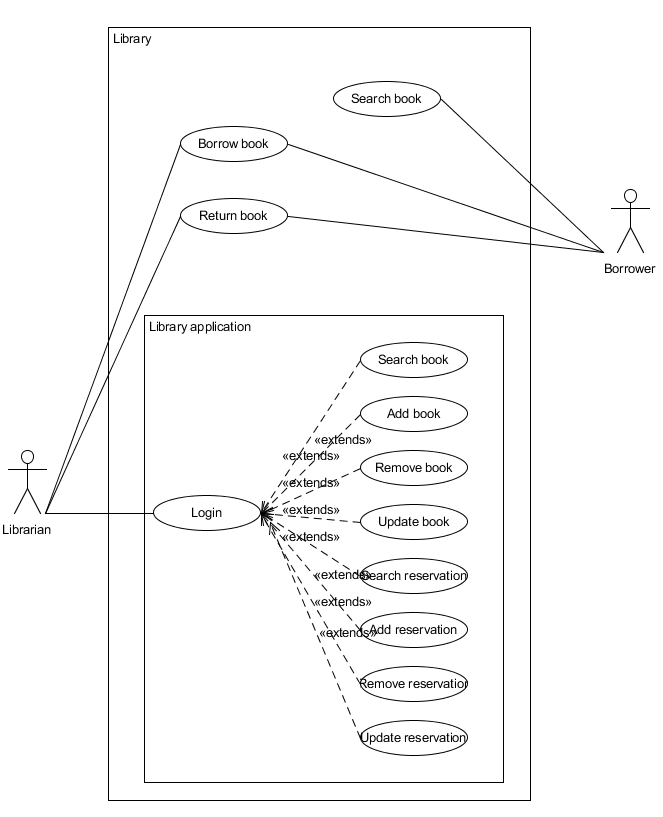
Database

*Figure 1 – Block diagram*

The Library application is what controls what’s in the system. Indeed, it defines the types of objects that are manipulated and who manipulates them, in our example the books and the administrators. It also will need to communicate with the database, sending information but also retrieving it for the user to consult. It will also have to be able to modify existing information in the database and update it.

## Main functionalities

Only one type of user interacts with the system: the librarian. In our application, he is considered as the administrator of the system, having the rights to manipulate every type of information that is present in the system: books, customers, reservations. The administrator can add a book to the library, define its type by selecting one of the available options, then entering the title, the author; he can add customers who have borrowed books by providing their first and last name; he can add a reservation for a certain book, providing the borrower’s identification, the borrow date and the return date, and determine if the book has been returned or not. He can also remove a book from the library or remove a customer from the database.



*Figure 2 – Use case diagram*

## Operating Environment

This application is ran using Java, so it is compatible with all operating systems. The current build is developed using Eclipse, with additional libraries for MySQL connections and plugins for the User Interface (javafx).

# System Features

## Project Features

Our take on this application is a classic one frequently used in data-centric projects: the MVC pattern. Consequently, the application is separated into three main packages: Model, Control/View, and Persistence.

Model is the central component, containing the application’s behavior domain independent of the user interface. It controls and manages the data of the system, as in what describes:

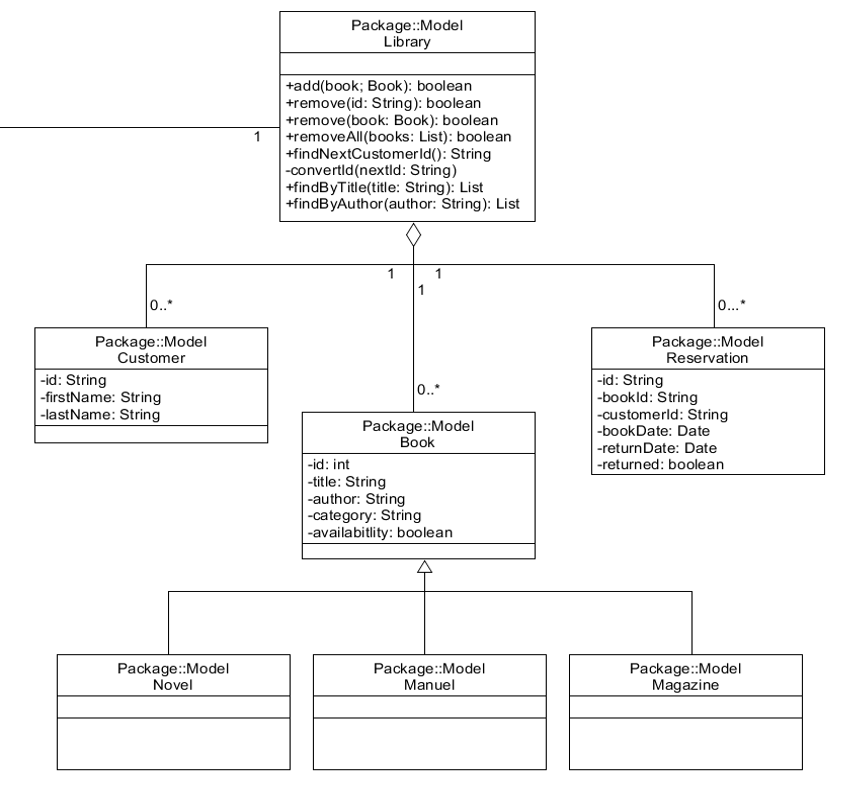
- a book (title, author, id etc.) and its type (whether it is a magazine, a novel or manual)

- an administrator (login credentials)

- a customer (first name, last name, id)

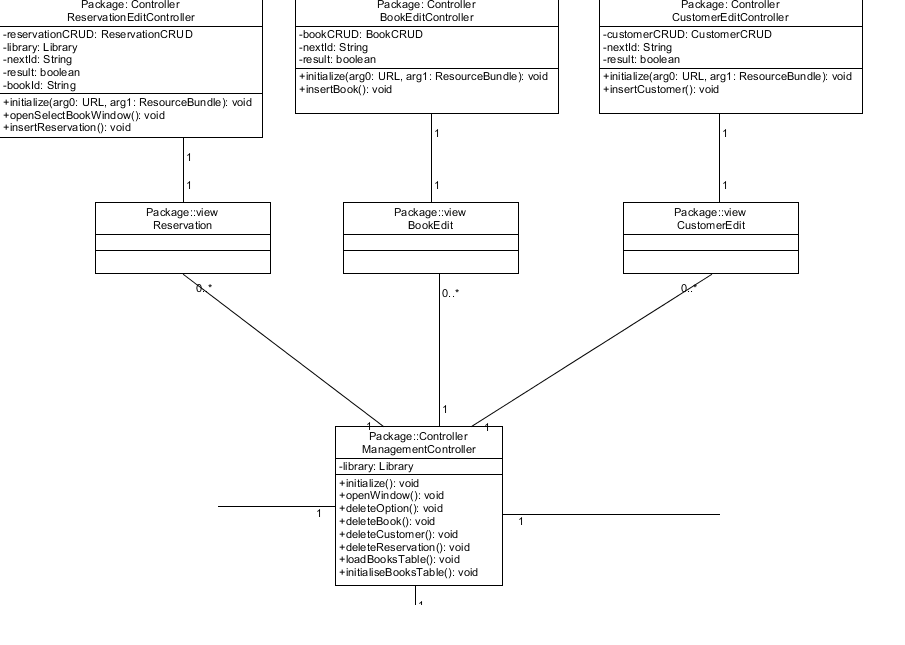
- a reservation (book id, customer id, id, book and return date, status as in if the book has been returned or not)

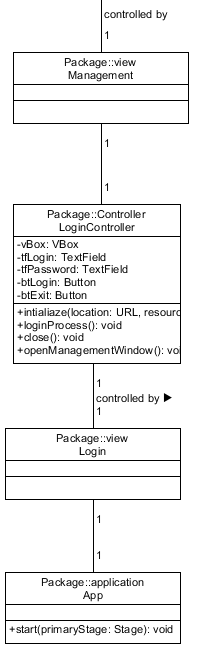
- what is in the library (array of books, customers and reservations)



*Figure 3 – Model domain model*

Control/View, named simply Controller, has the dual purpose of controlling the output representation of information (View) and managing input and converting it for the Model or View (Controller). As such, this is where the plugin javafx has been prominently used for handling the GUI. Each main action (select a book, edit a reservation, edit a customer, login) has a controller that manages accessors while also applying windows, tabs, tables, buttons and other main GUI components to said accessors, allowing user-interface interaction.

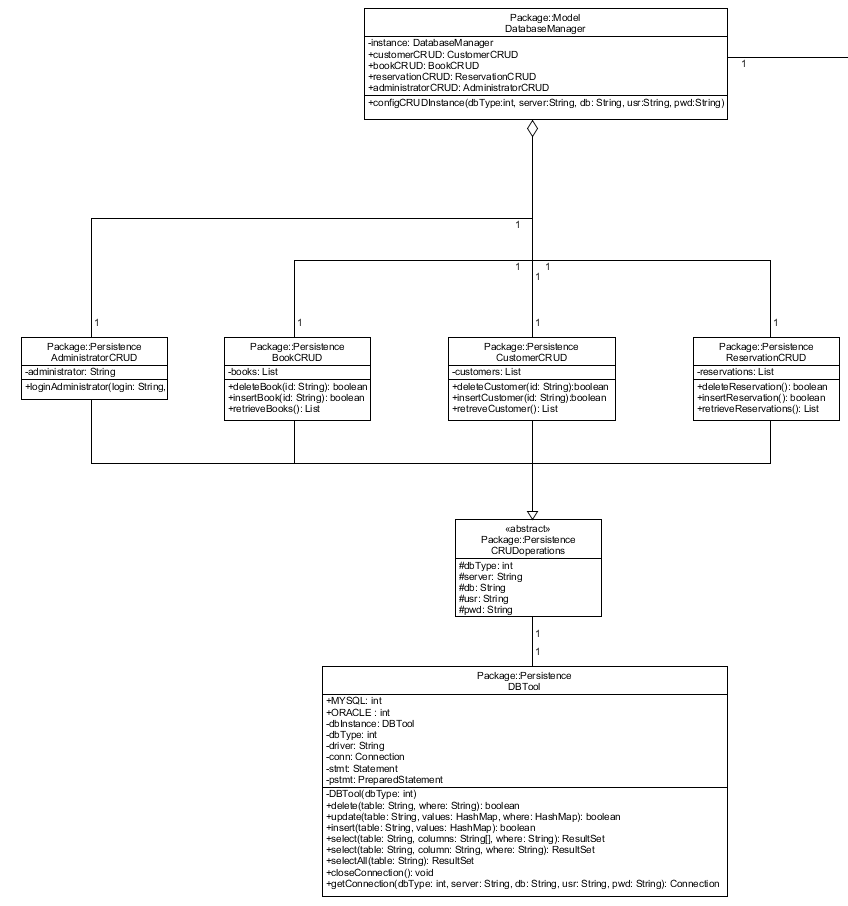




*Figure 4 – Controller domain model*

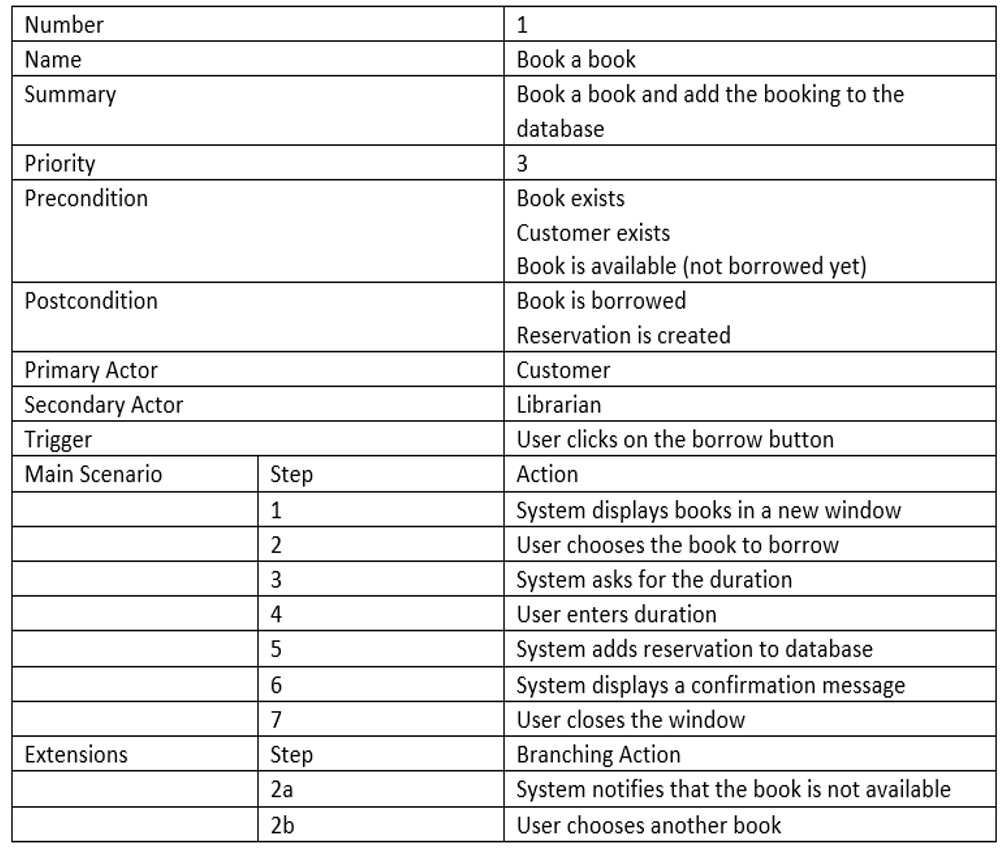
Persistence is the layer forming a direct relationship between the application and the database. Thanks to Persistence, data outlives process, meaning that it allows object and process characteristics to continue to exist even after the process that created it ceases or the machine it is running on is powered off. This is essential for improving energy consumption, as the client probably doesn’t intend to always leave the computer powered on for the library data to be stored. The user should be able to end all processes before leaving the library.

This package mainly uses CRUD operations to interact with the database, granting easy creation, reading, updating and deleting of residing objects.

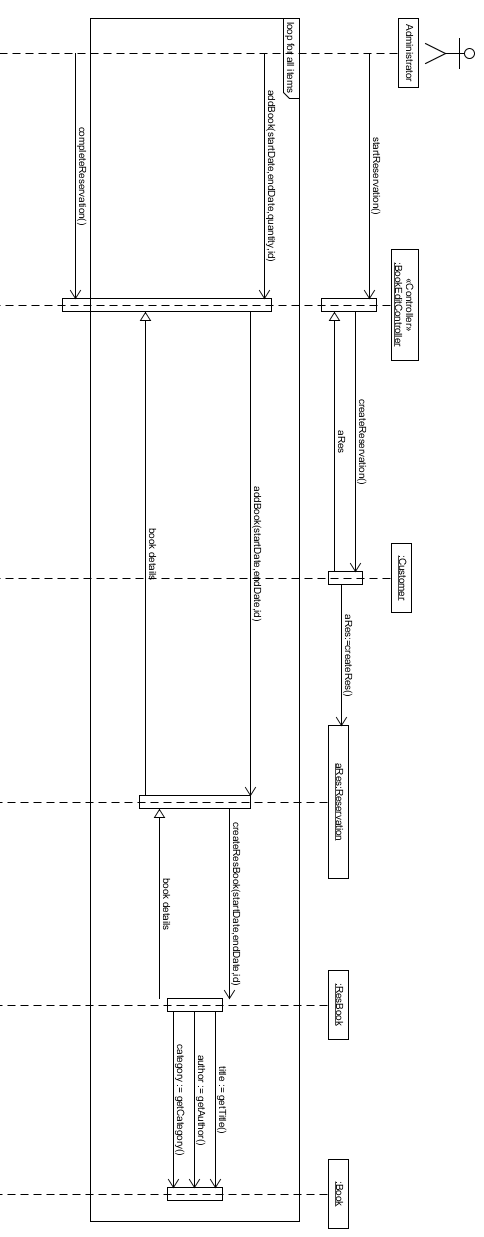


*Figure 5 – Persistence domain model*

## Use Case: make reservation



*Figure 7 – Scenario*

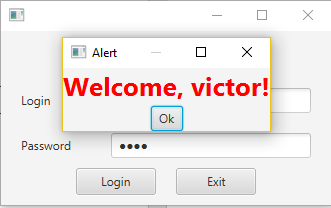


*Figure 8 – Sequence Diagram*

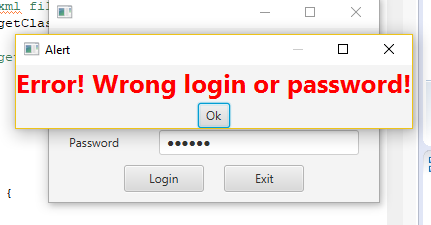
# External Interface Requirements

## User Interfaces

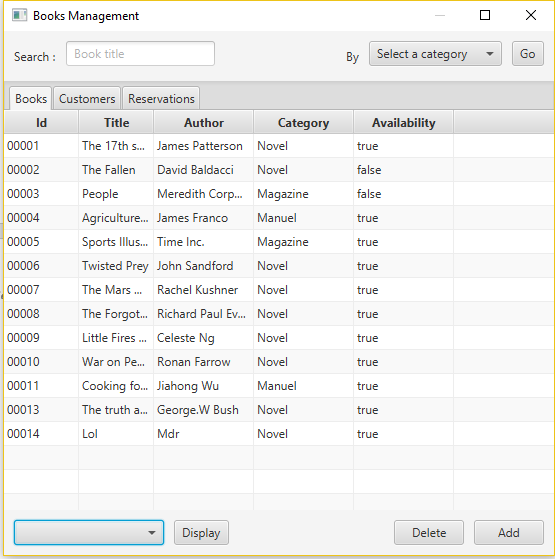
*Figure 9 – Login interface*



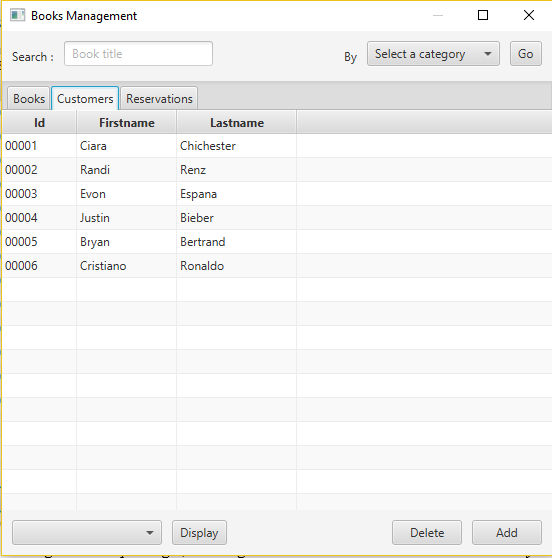
*Figure 10 – Login success*

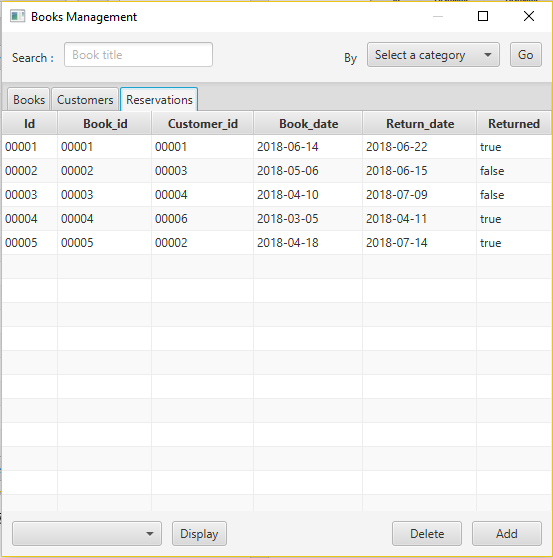


*Figure 11 – Login failed*

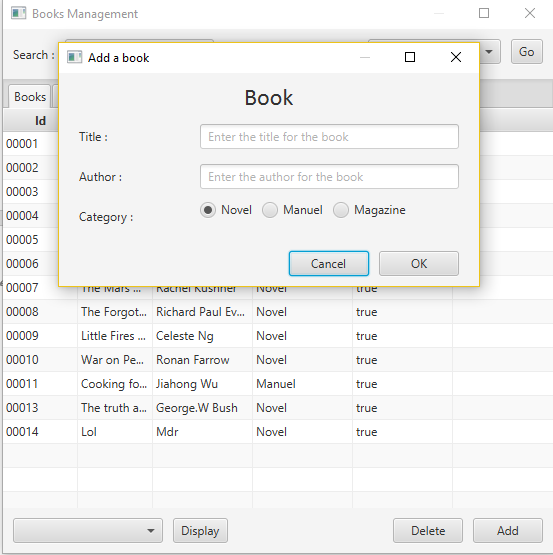


*Figure 12 – Upon login, we are greeted with the books available in the library*

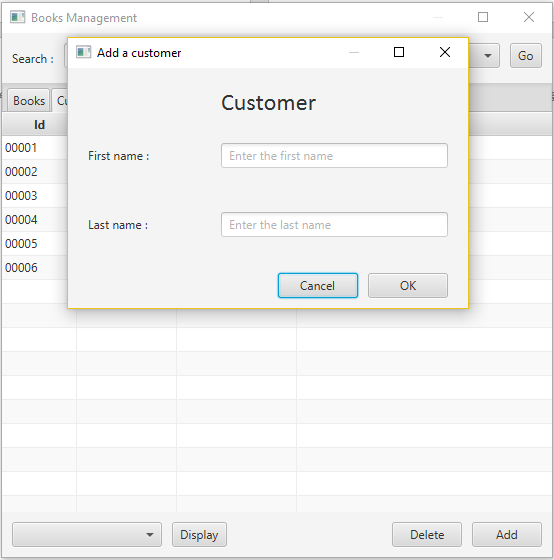




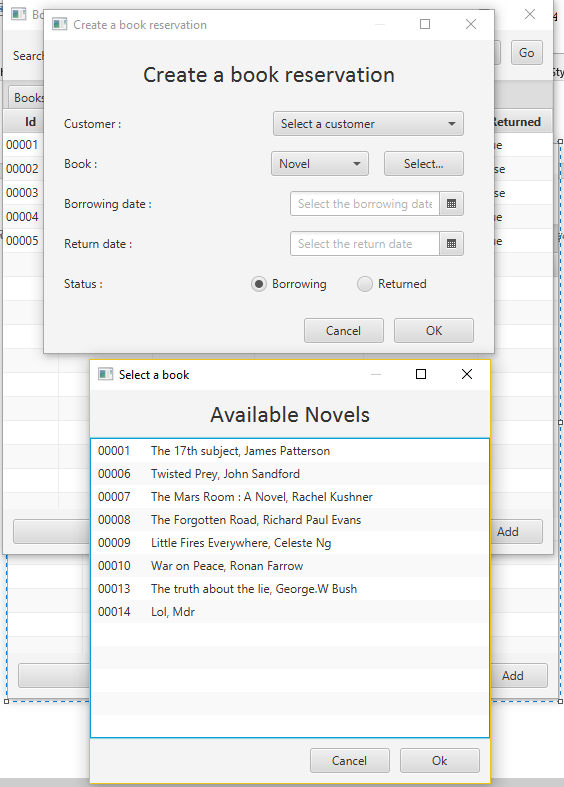
*Figure 13 – We can choose what we want to manage: books, customers or reservations*



*Figure 14 – Adding a book*



*Figure 15 – Adding a customer*



*Figure 16 – Creating a reservation*