Mercatino Libri Usati
Versione Digitale

RASD + DD + ITD

Jaunary 22, 2025
Version 1.0

Author: Elia Pontiggia
Mediator: Filippo Ambrosini

Revision History

Date	Revision	Notes
22/01/2025	v.0.0	Document creation
23/01/2025	v.1.0	First release

Contents

1 Introduction **4**
- 1.1 Purpose . . . 4
- 1.2 Scope . . . 4
- 1.3 Definitions, acronyms and abbreviations . . . 4

2 Requirement Analysis and Specification **5**
- 2.1 Hardware Interfaces . . . 5
- 2.2 Use Cases . . . 5
  - 2.2.1 Use Case 1: Book submission . . . 5
  - 2.2.2 Use Case 2: Book delivery . . . 6
  - 2.2.3 Use Case 3: Book sale . . . 7
  - 2.2.4 Use Case 4: Liquidation and return of unsold books . . . 7
- 2.3 Special Requirements . . . 7

3 Design **8**
- 3.1 Component diagram . . . 8
- 3.2 Logical description of data . . . 9
- 3.3 API endpoints . . . 9
  - 3.3.1 Login . . . 9
  - 3.3.2 Submit Books . . . 9
  - 3.3.3 Get Existing Books . . . 10
  - 3.3.4 Get Books in Stock . . . 10
  - 3.3.5 Enter book . . . 11
  - 3.3.6 Get PRs . . . 12
  - 3.3.7 Deliver Books . . . 12
  - 3.3.8 Sell Books . . . 13
  - 3.3.9 Liquidate PR and return unsold books . . . 13
  - 3.3.10 Mailing list . . . 13
- 3.4 Deployment . . . 13
- 3.5 UX design . . . 14
- 3.6 User interface . . . 14

4 Implementation **16**
4.1 Adopted development framework 16
4.2 Structure of the souece code 16
4.3 Test plan 17

A Appendix **18**
A.1 Database Creation Script 18

1. Introduction

1.1 Purpose

The purpose of the MLUD is to digitize and streamline the process of cataloging and monitoring the used schoolbook market held at Lokalino during the summer. The system aims to eeduce the workload on operators, particularly for repetitive tasks, minimize errors by eliminating manual transcription processes and significantly decrease paper usage.

1.2 Scope

The MLUD will be used by operators, providers and buyers. The system will allow operators to manage the catalog of books, providers to add books to the catalog and buyers to search for books. The system will also allow operators to account for the books sold and help the financial transactions between providers and buyers. The system does not handle actual financial transactions but keeps a record of the money involved in the transactions.
The boundaries of the system are as follows:

- Operations are limited to the Lokalino summer book market.
- The platform will only retain data for one year after market closure, except for users who opt into the mailing list.

1.3 Definitions, acronyms and abbreviations

Term	**Definition**	**Acronym**
Operator	A Lokalino employee	OP
Provider	A person who has books to sell	PR
Buyer	A person who wants to buy books	BY
Mercatino Libri	The system	MLUD

2. Requirement Analysis and Specification

The MLUD will offer the following functionalities:

- User registration and book detail submission via an online form.
- Facilitating the physical handover of books, including verification and labeling.
- Streamlining the sales process with a digital interface for managing transactions.
- Tracking unsold books and managing financial settlements for sellers.
- Generating aggregated statistical reports to monitor performance and user satisfaction.

2.1 Hardware Interfaces

The system will be accessible through a web interface. The PR will have the option to use both a computer and a mobile device to access the system, so the interface must be responsive and usable on both platforms.

The OP will use a computer to access the system, so the interface must be optimized for computer use.

2.2 Use Cases

The use cases are illustrated in the diagram in Figure 2.1.

2.2.1 Use Case 1: Book submission

1. The PR accesses the system.
2. The PR fills out the form, providing:
    - Personal information (name, surname, email, phone number).

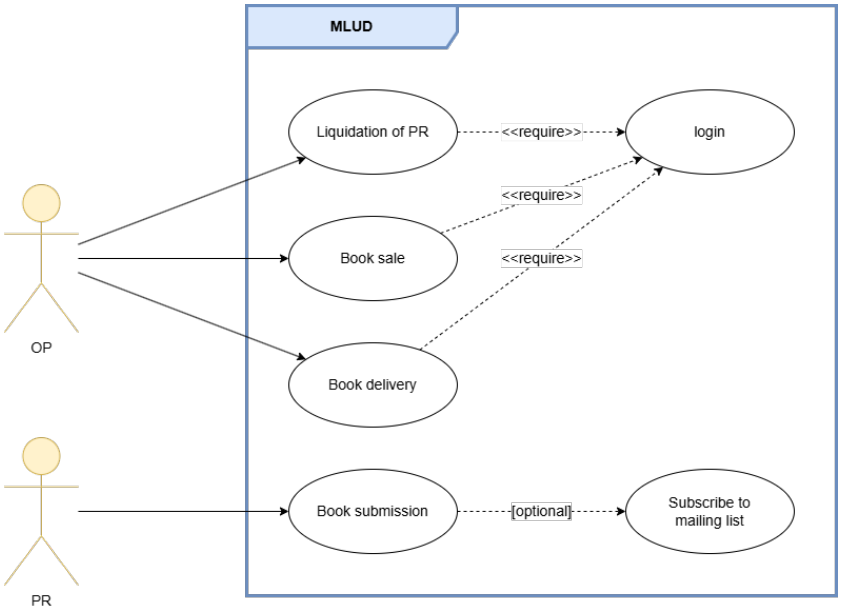

Figure 2.1: Use cases diagram

- Information for an arbitrary number of books (ISBN, title, author, edition, price, condition).
- Acceptance of the terms and conditions.
- Preference for subscription to the mailing list.

When entering the ISBN, the system will suggest the book's information if it is already in the system.

3. The PR submits the form, it is redirected to a confirmation page and receives a confirmation email.

2.2.2 Use Case 2: Book delivery

1. The PR goes to Lokalino with the books.
2. The OP accesses the protected area of the system.
3. The OP searches for the record of the PR in the system.
4. The OP verifies the correspondence between the books and the records, eventually updating the records or adding comments
5. If all is correct, the OP labels the books to identify the corresponding PR
6. The OP mark the correct delivery of the books in the system

2.2.3 Use Case 3: Book sale

1. The OP accesses the protected area of the system.
2. The BY selects the books they want to buy.
3. For each book, the OP searches for the record in the system and adds the book to the cart.
4. At the end of the selection, the OP confirms the purchase and collects the payment. The books are marked as sold in the system.

2.2.4 Use Case 4: Liquidation and return of unsold books

1. The OP accesses the protected area of the system.
2. The OP searches for the record of the PR in the system.
3. The system shows the list of books unsold and the total amount due to the PR.
4. The PR collects the money and the unsold books.
5. The OP marks the books as returned in the system.

2.3 Special Requirements

In order to facilitate the filling of the form by the PR, the system will provide a search function that will suggest the book's information when the ISBN is entered. In order to generate this suggestion, the system will also allow the OP to manually enter the book's information, possibly before the beginning of the event.

The system will keep the data of the PRs and the books for 1 year after the event, and after that, it will be deleted. The only data that will be kept indefinitely is the list of persons who have subscribed to the mailing list.

3. Design

3.1 Component diagram

The system will be organized in a classical client-server configuration, and the latter will expose RESTful APIs to the clients, and a web application for both OPs and PRs. Every service will be in its own servlet. The component diagram is shown in Figure 3.1.

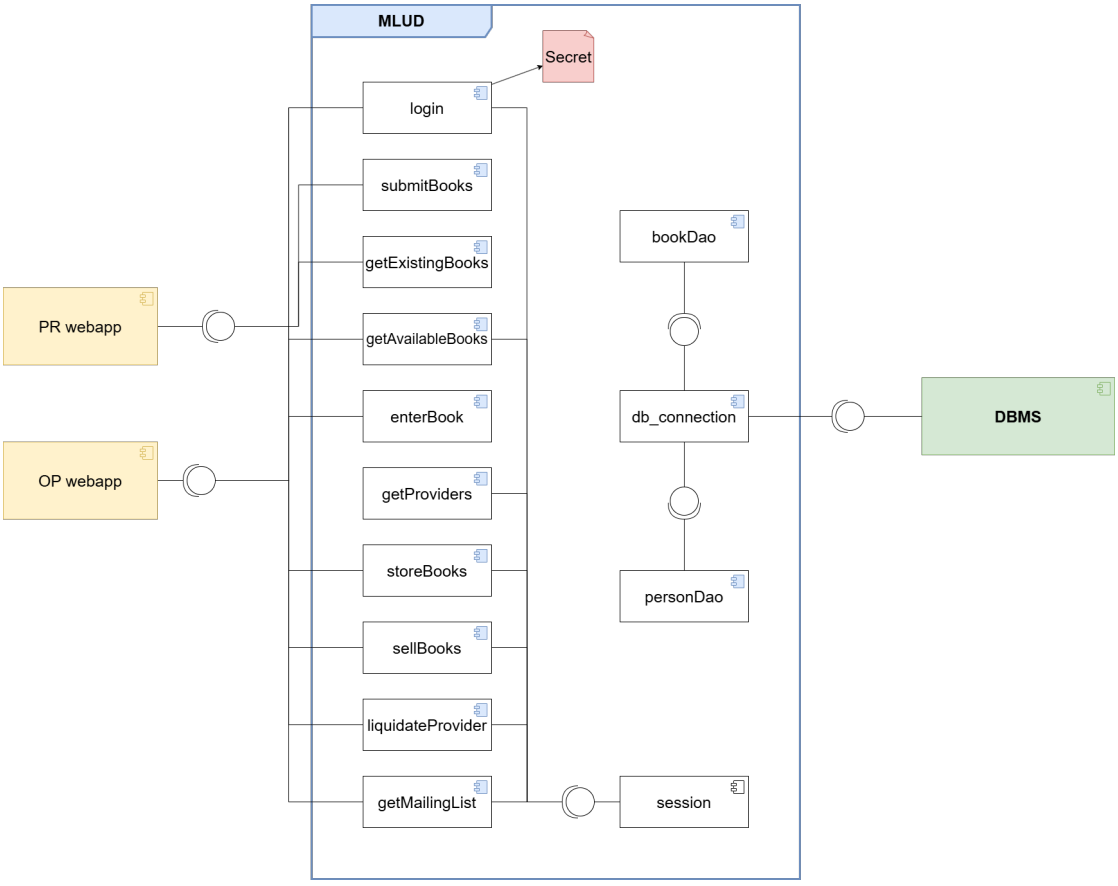

Figure 3.1: Component diagram

3.2 Logical description of data

The data will be stored into a relational database, which will be designed according to the Logical schema in Figure 3.2.

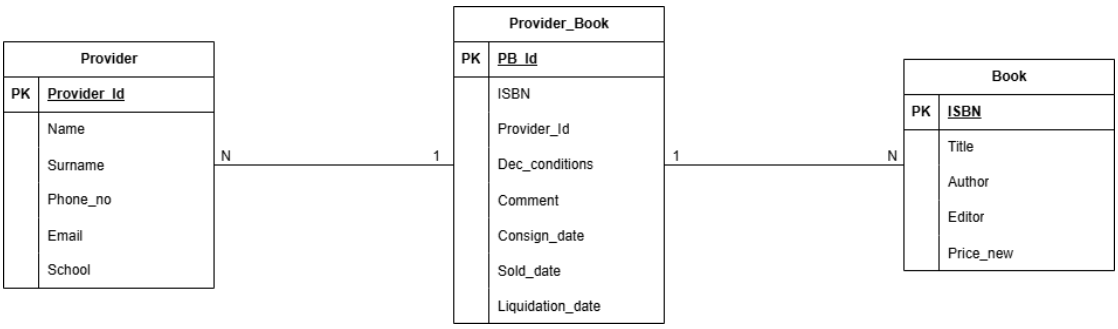

Figure 3.2: Logical schema of the database

3.3 API endpoints

The API will expose the endpoints in a structure coherent with the compoenent diagram 3.1. It follows a list of the endpoints:

3.3.1 Login

Method : POST
EndPoint : /login.php
Body parameters :

```
{ "Hashed_password": String }
```

Response : `200` if the login is successful, `401` if the password is wrong.

3.3.2 Submit Books

A PR submits its books to the system, and gives its personal information.
Method : POST
EndPoint : /submitBooks.php
Body parameters :

```
{
    "Name": String,
    "Surname": String,
    "School": String,
    "Email": String,
    "Phone": String,
```

```
    "Books": [
        {
            "ISBN": String,
            "Title": String,
            "Author": String,
            "Editor": String,
            "Price_new": Number,
            "Dec_conditions": String
        },
        ...
    ],
    "Mail_list": Boolean // TODO: actually add it into the
       database and the form
}
```

Response : `200` if the submission is successful, `400` if the request is malformed.

3.3.3 Get Existing Books

The system returns the information of the books already in the system, searching by ISBN.
Method : GET
EndPoint : /getExistingBooks.php
URL parameters :

- `ISBN`: String

Response :

```
[
    {
        "ISBN": String,
        "Title": String,
        "Author": String,
        "Editor": String,
        "Price_new": Number
    },
    ...
]
```

3.3.4 Get Books in Stock

The system returns all the books in stock, i.e. the books both submitted and actually delivered by the PRs.
Method : GET

EndPoint : /getAvailableBooks.php
Response :

```
[
    {
        "ISBN": String,
        "Title": String,
        "Author": String,
        "Editor": String,
        "Price_new": Number,
        "ProviderName": String,
        "ProviderSurname": String,
        "PB_Id": Number,
        "Provider_Id": Number,
        "Dec_conditions": String,
        "Comments"? : String,
        "Consign_date": String,
        % // didn't have the sbatta to remove them from the
           select
        "Sold_date": null,
        "Liquidation_date": null

    },
    ...
]
```

3.3.5 Enter book

An OP enters a book in the system, associating it to a ISBN. No strong need for this, it's just to make simpler the PR submission.
Method : POST
EndPoint : /enterBook.php
Body parameters :

```
{
    "ISBN": String,
    "Title": String,
    "Author": String,
    "Editor": String,
    "Price_new": Number
}
```

Response : `200` if the submission is successful, `400` if the request is malformed.

3.3.6 Get PRs

The system returns the list of PRs, to handle the delivery of the books and the liquidation.
Method : GET
EndPoint : /getProviders.php
Response :

```
[
    {
        "ISBN": String,
        "Title": String,
        "Author": String,
        "Editor": String,
        "Price_new": Number,
        "Name": String,
        "Surname": String,
        "PB_Id": Number,
        "Provider_Id": Number,
        "Dec_conditions": String,
        "Comments"? : String,
        "Consign_date"?: String,
        "Sold_date"?: String,
        "Liquidation_date"?: String,
        "Phone_no": String,
        "Email": String,
        "School": String
    },
    ...
]
```

3.3.7 Deliver Books

A PR delivers the books to the OP.
Method : POST
EndPoint : /storeBooks.php
Body parameters :

```
[
    {
        "PB_Id": Number,
        "Comments"?: String
    },
    ...
]
```

Response : `200` if the submission is successful, `400` if the request is malformed.

3.3.8 Sell Books

An OP sells the books to a BY.
Method : POST
EndPoint : /sellBooks.php
Body parameters :

```
[
    { "PB_Id": Number },
    ...
]
```

Response : `200` if the submission is successful, `400` if the request is malformed.

3.3.9 Liquidate PR and return unsold books

An OP liquidates the PR, returning the unsold books and the money.
Method : POST
EndPoint : /liquidateProvider.php
Body parameters :

```
{ "Provider_Id": Number, }
```

Response : `200` if the submission is successful, `400` if the request is malformed.

3.3.10 Mailing list

Show the list of PRs that accepted to be in the mailing list.
Method : GET
EndPoint : /mailingList.php
Response :

```
[
    {
        "Name": String,
        "Surname": String,
        "Email": String
    },
    ...
]
```

3.4 Deployment

All the webapp will be entirely deployed on the free hosting service Altervista, which provides a MySQL database and a php server, so all the problems of deployment, reliability, scalability, and security are handled by the service provider.

3.5 UX design

The webapp will be designed with a simple and intuitive interface, with a clean and modern look.

The UX is explained at an high level in the flow diagrams in Figures 3.3 and 3.4.

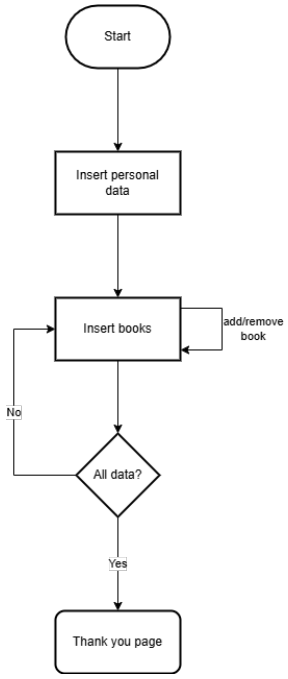

Figure 3.3: Flow of the book submission

3.6 User interface

Still WIP. Screenshots will be added in a future release of the document.

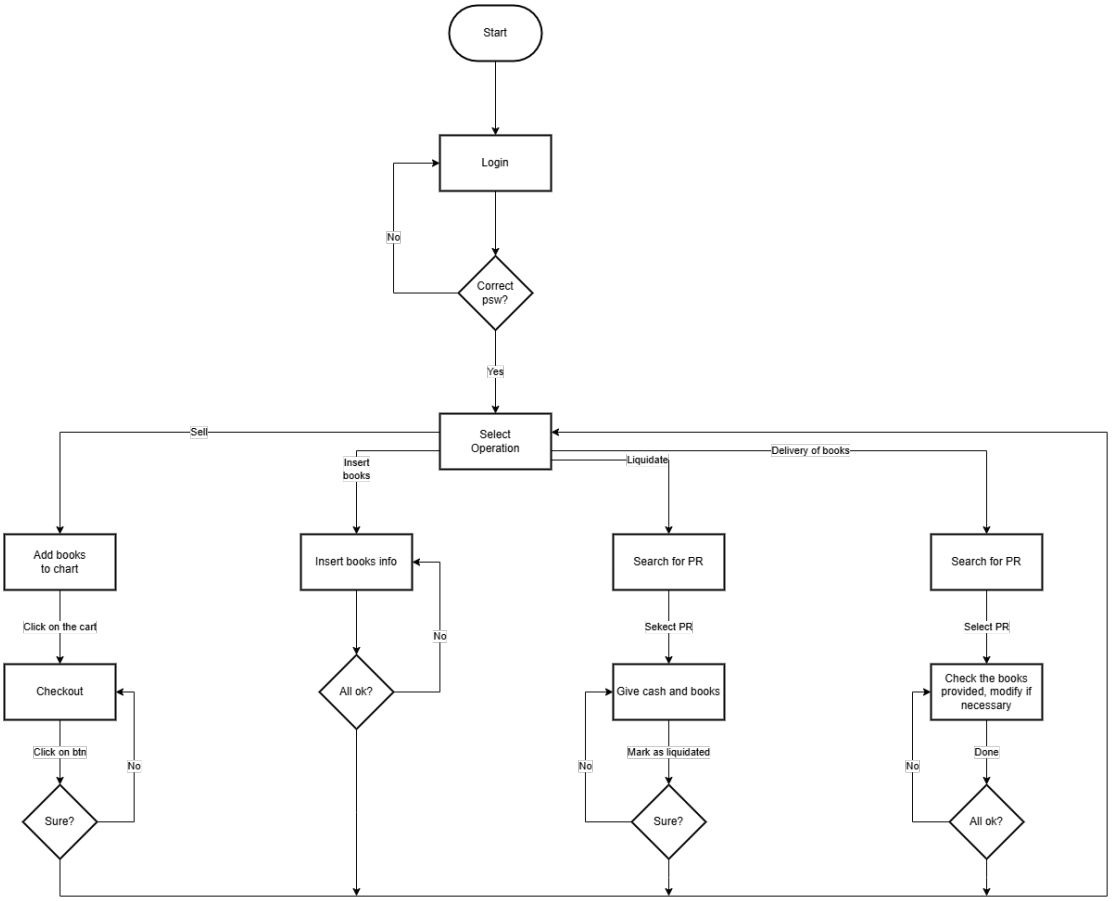

Figure 3.4: Flow of the OP

4. Implementation

All the code is open source under the Apache 2.0 license and can be found on GitHub at

`https://github.com/pontig/lokalino_mlud`.

4.1 Adopted development framework

Backend

The backend is implemented in **php**, since it is the language that the author is most familiar with and it is the only backend supported by Altervista (the free hosting service used for the project).

Frontend

The frontend is implemented in **React**. React is a JavaScript library for building user interfaces. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications, since it is capable of handling the view layer for web and mobile apps. React was chosen because it is a modern and widely used library that allows for the creation of a dynamic and responsive user interface.

4.2 Structure of the souece code

The source code is structured in a way that separates the frontend from the backend, both organized in an intuitive way and easy to scale and maintain.

The full structure of the source code is as follows:

`backend/` - contains the backend code

- `dao/` - contains the data access objects, one for the books and one for the PR. it also contains the database connection
- `utils/` - contains the utility functions, such as the session management and the secret passwords of the OPs
- `*.php` - the servlets that handle the requests from the frontend

`frontend/` - contains the React application

- `build/` - contains the compiled code (not included in the repository)
- `public/` - contains the static files, such as the index.html and the icons
- `src/` - contains the source code
  - `pages/` - contains the pages of the application
  - `styles/` - contains the stylesheets
  - `types/` - contains the typescript interfaces
  - `App.js` - the main component of the application
  - `index.js` - the entry point of the application

4.3 Test plan

The testing of the application will be done once the development is complete. The testing will be done manually, by the OPs, simulating a real user experience.

All this process will be supervised by the author of the project.

A. Appendix

A.1 Database Creation Script

Listing A.1: Database Creation Script

```
-- Table: Provider
CREATE TABLE Provider (
    Provider_Id INT PRIMARY KEY NOT NULL,
    Name VARCHAR(100) NOT NULL,
    Surname VARCHAR(100) NOT NULL,
    Phone_no VARCHAR(15) NOT NULL,
    Email VARCHAR(100) NOT NULL
);

-- Table: Book
CREATE TABLE Book (
    ISBN CHAR(13) PRIMARY KEY NOT NULL,
    Title VARCHAR(200) NOT NULL,
    Author VARCHAR(100) NOT NULL,
    Editor VARCHAR(100) NOT NULL,
    Price_new DECIMAL(10, 2) NOT NULL
);

-- Table: Provider_Book (Linking Table)
CREATE TABLE Provider_Book (
    PB_Id INT PRIMARY KEY NOT NULL,
    ISBN CHAR(13) NOT NULL,
    Provider_Id INT NOT NULL,
    Dec_conditions VARCHAR(100) NOT NULL,
    Comment VARCHAR(200),
    Consign_date TIMESTAMP DEFAULT NULL,
    Sold_date TIMESTAMP DEFAULT NULL,
    Liquidation_date TIMESTAMP DEFAULT NULL,
    FOREIGN KEY (ISBN) REFERENCES Book(ISBN),
    FOREIGN KEY (Provider_Id) REFERENCES Provider(Provider_Id)
);
```