



Mercatino Libri Usati  
Versione Digitale

RASD + DD + ITD

January 22, 2025  
Version 3.0

<i>Author:</i>	Elia Pontiggia
<i>Mediator:</i>	Filippo Ambrosini

# Revision History

Date	Revision	Notes
22/01/2025	v.0.0	Document creation
23/01/2025	v.1.0	First release
31/01/2025	v.2.0	Deployment of the first version of the system

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Purpose . . . . .	4
1.2	Scope . . . . .	4
1.3	Definitions, acronyms and abbreviations . . . . .	4
<b>2</b>	<b>Requirement Analysis and Specification</b>	<b>5</b>
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
2.4	Assumptions . . . . .	8
<b>3</b>	<b>Design</b>	<b>9</b>
3.1	Component diagram . . . . .	9
3.2	Logical description of data . . . . .	10
3.3	API endpoints . . . . .	10
3.3.1	Login . . . . .	10
3.3.2	Submit Books . . . . .	10
3.3.3	Get Existing Books . . . . .	11
3.3.4	Get Books in Stock . . . . .	11
3.3.5	Enter book . . . . .	12
3.3.6	Get PRs . . . . .	13
3.3.7	Get Books submitted by a PR . . . . .	13
3.3.8	Deliver Books . . . . .	14
3.3.9	Sell Books . . . . .	15
3.3.10	Get Situation for a PR after delivery . . . . .	15
3.3.11	Liquidate PR and return unsold books . . . . .	15
3.3.12	Mailing list . . . . .	16
3.3.13	Session check . . . . .	16
3.3.14	Logout . . . . .	16

---

3.3.15	Get Schools . . . . .	16
3.3.16	Get Book Adopted by a School . . . . .	17
3.4	Deployment . . . . .	17
3.5	UX design . . . . .	18
3.6	User interface . . . . .	18
<b>4</b>	<b>Implementation</b>	<b>24</b>
4.1	Adopted development framework . . . . .	24
4.2	Structure of the source code . . . . .	24
4.3	Test plan . . . . .	25
<b>A</b>	<b>Database Creation Script</b>	<b>27</b>
<b>B</b>	<b>Changelog</b>	<b>28</b>
B.1	Version 1.0 to 2.0 . . . . .	28
B.2	Version 2.0 to 3.0 . . . . .	28

# 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 reduce 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, school of origin).



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.
- Acceptance of Lokalino rules.
- Preference for subscription to the mailing list.

The system will allow the PR to fill out the book information after the school of origin, so that the suggestion of the book's information can be generated based on the schools that adopted the books.

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 generate the list of books that can be suggested to the PR, the system will provide the possibility to the OP to insert the list of books that have been adopted by the schools in the area. The OP will be able to insert the list of books in a facilitated way, to be defined directly with the OP a short time before the delivery.
- 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.
- Every time an OP handles information about a PR, the system will also show the unique ID of the PR, so that the OP can easily label and identify the books.



## **2.4 Assumptions**

An assumptions have been made in the design of the system:

Even if in the implementation every PR will be identified by a unique ID, the system will check the uniqueness of the email address; this comes into play when a PR submits the form: if the email address is already in the system and the PR fills out the form again, the system will not create a new record and will work as if the same PR is inserting more books.

## 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.

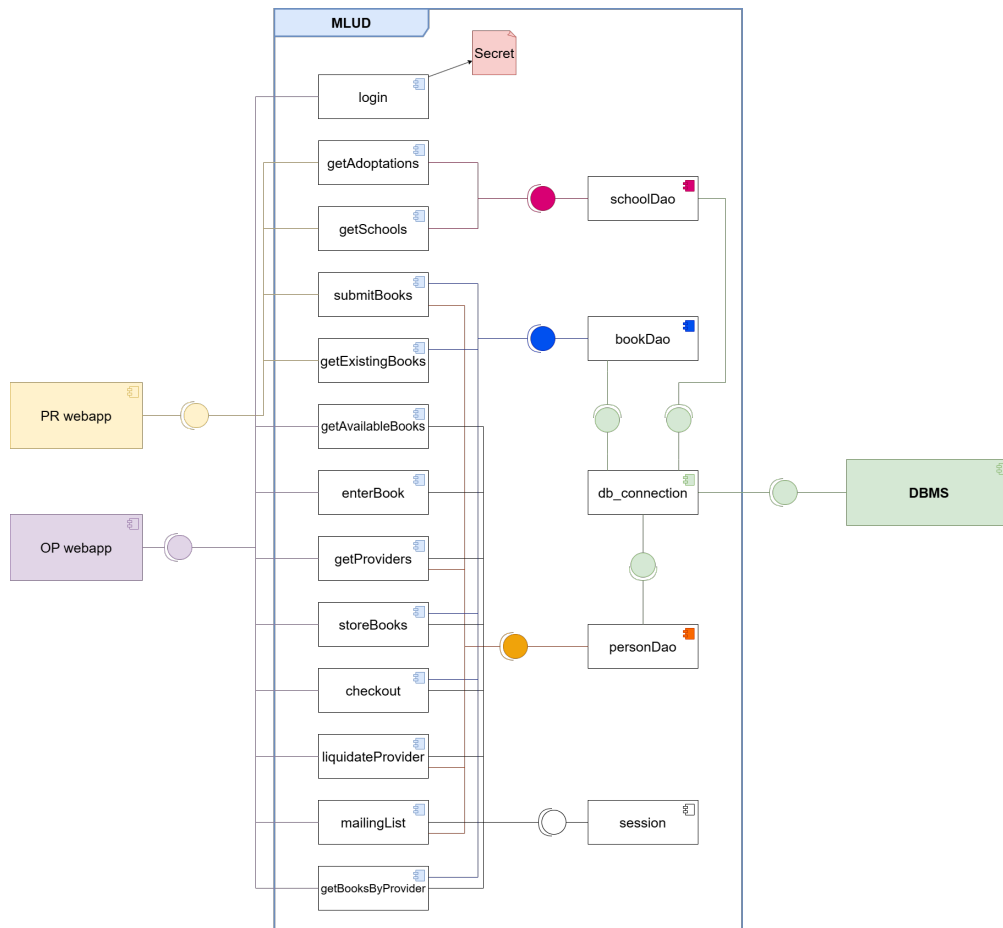


Figure 3.1: Component diagram

## 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.

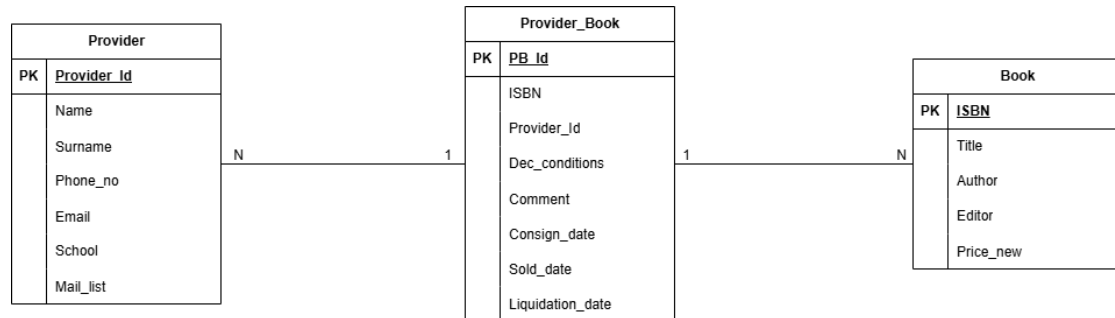


Figure 3.2: Logical schema of the database

## 3.3 API endpoints

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

### 3.3.1 Login

**Method :** POST

**EndPoint :** /login.php

**Body parameters :**

```
{ "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 :**

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

```
        "Phone_no": String,  
        "Mail_list": Boolean  
    },  
    "books": [  
        {  
            "ISBN": String,  
            "Title": String,  
            "Author": String,  
            "Editor": String,  
            "Price_new": Number,  
            "Dec_conditions": String  
        },  
        ...  
    ]  
}
```

**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 or Title, depending on the parameters.

**Method :** GET

**EndPoint :** /getExistingBooks.php

**URL parameters :**

- ISBN: String
- Title: 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,
    "Comment"? : String,
    "Consign_date": String,
    "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. **Method** : GET

**EndPoint** : /getProviders.php

**Response** :

```
[
  {
    "Provider_Id": Number,
    "Name": String,
    "Surname": String,
    "State": String
  },
  ...
]
```

The possible states are:

- 0 - waiting for delivery
- 1 - waiting for liquidation
- 2 - liquidated

### 3.3.7 Get Books submitted by a PR

The system returns the books submitted by a PR. Useful for book delivery

**Method** : GET

**EndPoint** : /getBooksByProvider.php

**URL parameters** :

- **Provider\_Id**: Number

**Response** :

```
{
  "books": [
    {
      "PB_Id": Number,
      "ISBN": String,
      "Title": String,
      "Author": String,
      "Editor": String,
      "Price_new": Number,
      "Dec_conditions": String,
      "Sold_date": Timestamp
    },
    ...
  ]
}
```

```
    ],
    "provider": [
        {
            "Name": String,
            "Surname": String
        }
    ]
}
```

### 3.3.8 Deliver Books

A PR delivers the books to the OP.

**Method** : POST

**EndPoint** : /storeBooks.php

**Body parameters** :

```
{
    "Provider_Id": Number,
    "Books_to_edit": [
        {
            "PB_Id": Number,
            "Dec_conditions": String,
            "Comment"?: String
        },
        ...
    ],
    "Books_to_add": [
        {
            "ISBN": String,
            "Title": String,
            "Author": String,
            "Editor": String,
            "Price_new": Number,
            "Dec_conditions": String,
            "Comment"?: String
        },
        ...
    ],
    "Books_to_remove": [
        { "PB_Id": Number },
        ...
    ]
}
```

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

### 3.3.9 Sell Books

An OP sells the books to a BY.

**Method** : POST

**EndPoint** : /checkout.php

**Body parameters** :

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

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

### 3.3.10 Get Situation for a PR after delivery

The system returns the situation of a PR after the delivery: unsold books and how much money the PR has to receive.

**Method** : GET

**EndPoint** : /getSituation.php

**URL parameters** :

- **Provider\_Id**: Number

**Response** :

```
{
    "Name": String,
    "Surname": String,
    "Unsold_books": [
        {
            "PB_Id": Number,
            "ISBN": String,
            "Title": String,
            "Author": String,
            "Editor": String,
            "Price_new": Number,
            "Dec_conditions": String
        },
        ...
    ],
    "Money": Number
}
```

### 3.3.11 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.

### 3.3.12 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,
    "Phone_no": String,
    "Email": String,
    "School": String,
    "Mail_list": Boolean
  },
  ...
]
```

### 3.3.13 Session check

Check if the session is still valid.

**Method** : GET

**EndPoint** : /utils/session.php

**Response** : **200** if the session is still valid, **401** if the session is expired.

### 3.3.14 Logout

Logout the OP.

**Method** : GET

**EndPoint** : /logout.php

**Response** : **200** if the logout is successful.

### 3.3.15 Get Schools

When a PR starts to fill the form, it has to select its school of origin. The system returns the list of schools.

**Method** : GET

**EndPoint** : /getSchools.php

**Response** :

```
[
  {
    "School_Id": Number,
    "Name": String,
    "Is_HighSchool": Boolean
  },
  ...
]
```

### 3.3.16 Get Book Adopted by a School

When a PR fills the "school of origin" field, the system suggests the books adopted by that school.

**Method** : GET

**EndPoint** : /getAdoptedBooks.php

**URL parameters** :

- **School\_Id**: Number

**Response** :

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

## 3.4 Deployment

All the webapp will be entirely deployed on the free hosting service Alservista, 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.

Particularly, the link to the webapp will be <https://lokalinomlud.altervista.org/>

### 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.

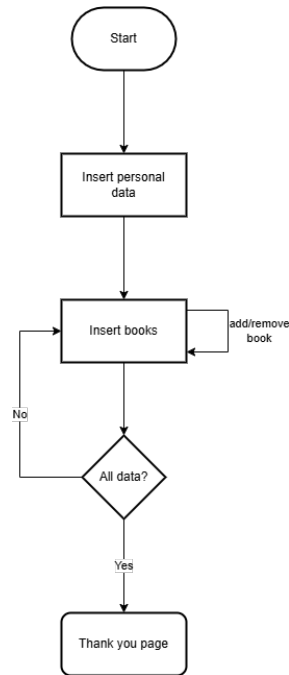


Figure 3.3: Flow of the book submission

### 3.6 User interface

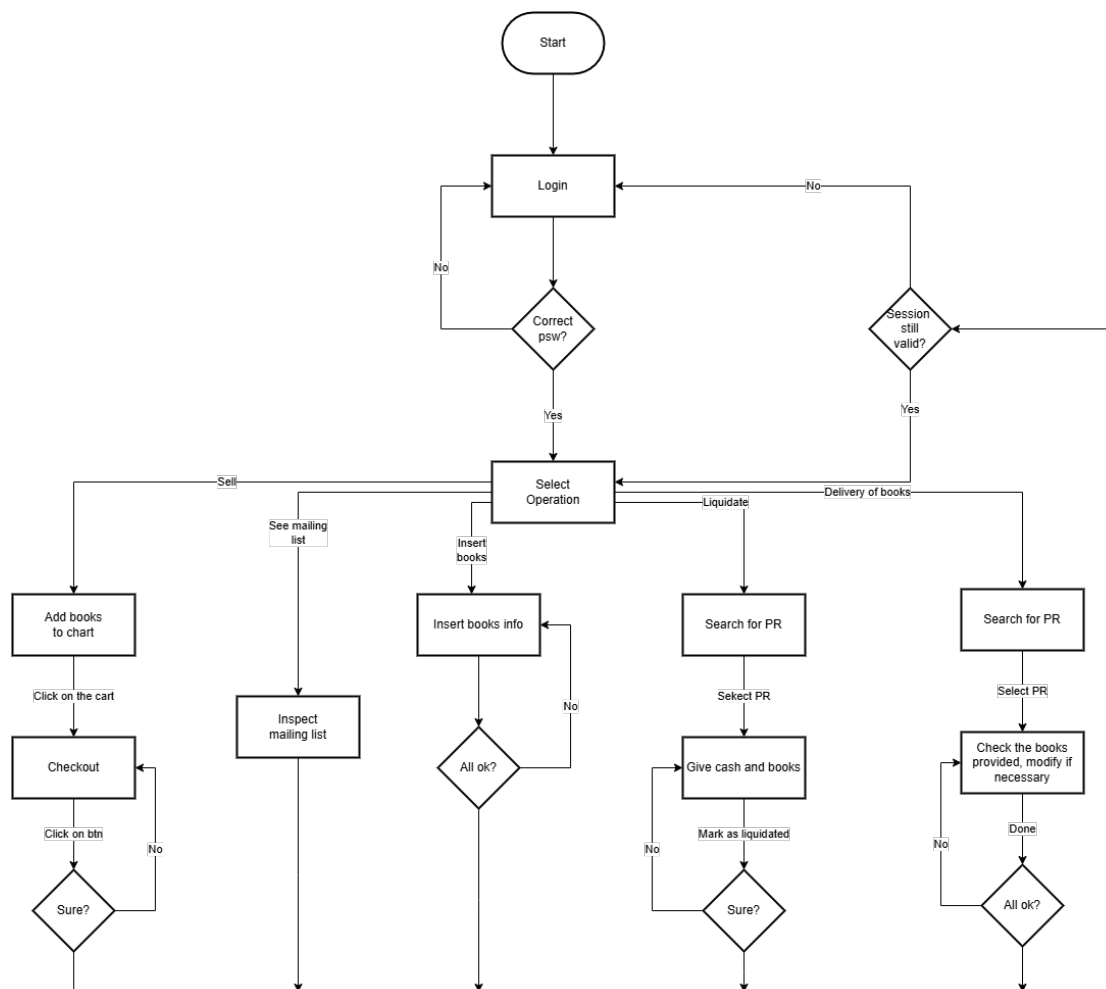


Figure 3.4: Flow of the OP

**Login al nuovo merkatino libri usati digitale**

Login

Figure 3.5: Login page

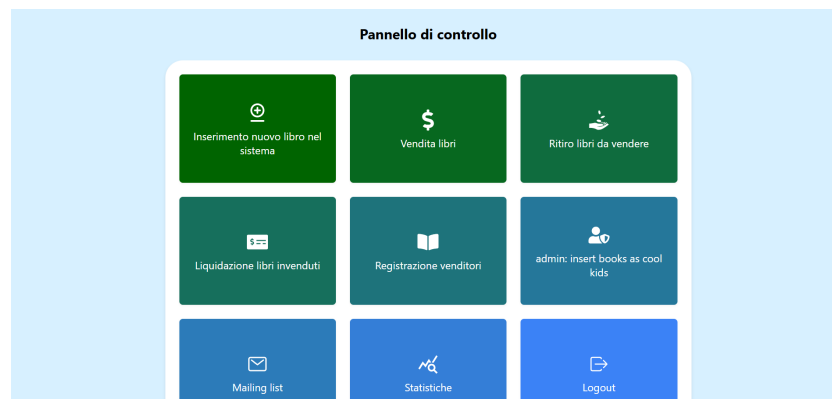


Figure 3.6: Dashboard

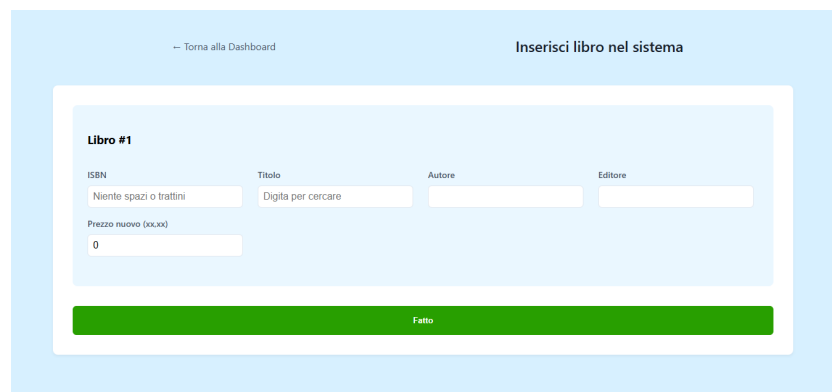


Figure 3.7: Insert book into the system



Figure 3.8: Select a PR to pick up its books

— Seleziona un altro venditore

Controlla libri portati da Elia Pontiggia

**Libri dichiarati**

**Libro #1** Rimuovi

ISBN	Titolo	Autore	Editore
9781380048325	Get Inside Grammar - English A1	AA.VV.	Macmillan
Prezzo nuovo (v.u.)	Condizioni	Aggiungi un commento se necessario	
34,60	Molto usurato	Es. Copertina rovinata	

**Libro #2** Rimuovi


ISBN	Titolo	Autore	Editore
9788024707328	ABC Delle Scienze Motorie + Lib	Balboni G.	Il Capitello
Prezzo nuovo (v.u.)	Condizioni	Aggiungi un commento se necessario	
22,00	Buono	Es. Copertina rovinata	

Figure 3.9: List of books to pick up from a PR

**Selezione libri**

— Pannello di controllo

Cerca libri...



**Riflessi Narrativa Poesia Teatro**  
di Daniele, Franz  
Editore: Loescher Editore  
Venduto da Sonia Barini, stato average  
€19.80 Rimuovi

**Narrami O Musa Volume Unico**  
di Ciocca Daniela, Ferri Tina  
Editore: A. Mondadori Scuola  
Venduto da Sonia Barini, stato average  
€31.40 Nel carrello

**Il Nuovo Segni Dei Tempi - Volume Corso Di Religione Cristianesimo In Dialogo Col Mondo**  
di Pasquale, Panzoli  
Editore: La Scuola Editrice  
Venduto da Giovanni Giacomo, stato bad  
govhgvhg  
€18.90 Rimuovi

**Chimica Volume 1 + Ebook + Workbook Per Il Ripasso E Il Recupero**  
di Ricci Giovanni, De Leo Marinella  
Editore: De Agostini

**Disegno Teoria E Realtà Volume Unico + Laboratorio**  
di Dorflès Gillo, Lazzaretti Tiziana, Pinotti Annibale  
Editore: Atlas  
Venduto da Pippo Giovanni, stato average

**Risposte Della Fisica (Le) Volume 1° Anno**  
di Caforio Antonio, Ferilli Aldo  
Editore: Le Monnier  
Venduto da Pippo Giovanni, stato good

Figure 3.10: Books to sell

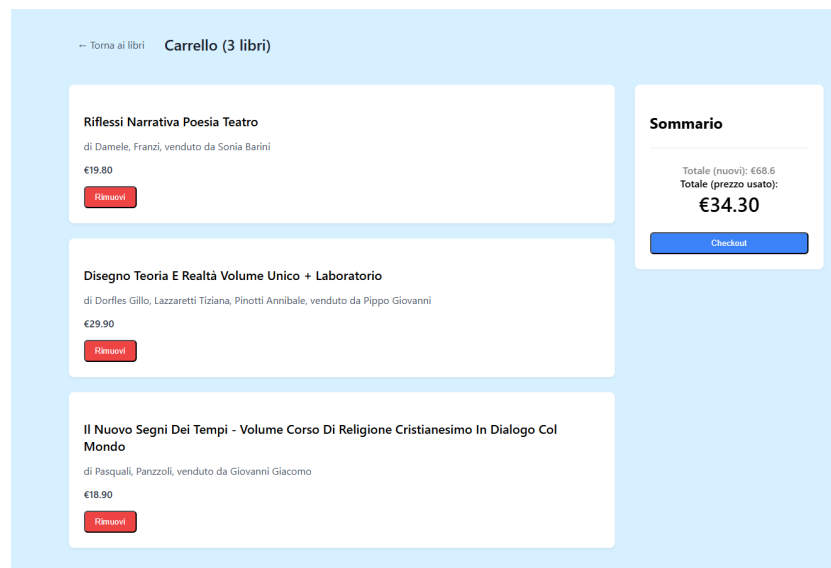


Figure 3.11: Cart

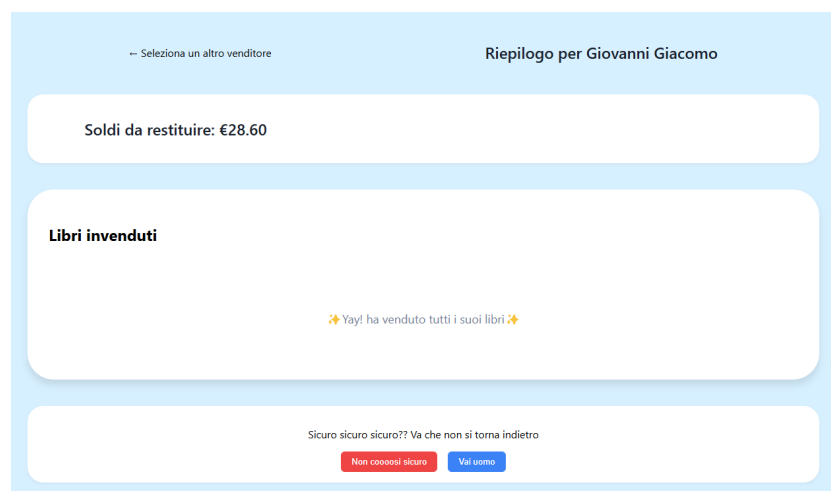


Figure 3.12: Liquidation of a PR



Figure 3.13: Mailing list

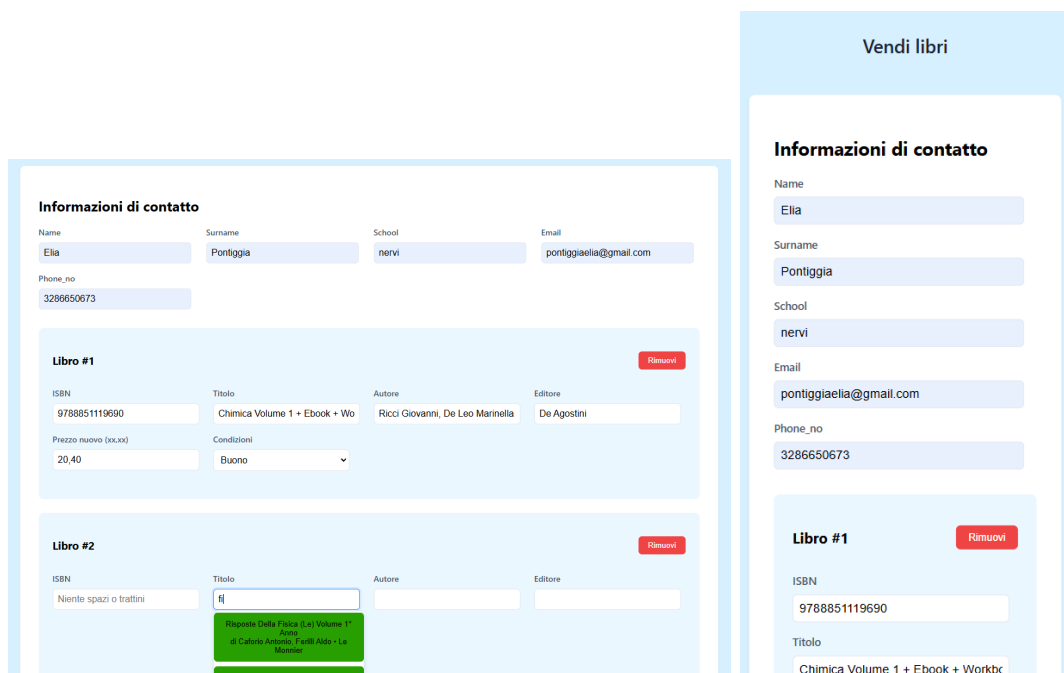


Figure 3.14: PR registration in both scenarios: from a PC and from a smartphone



## 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](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 Altvista (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 source 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

**components/** - contains the reusable components

**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.

# Appendix

## A. Database Creation Script

Listing A.1: Database Creation Script

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

## B. Changelog

### B.1 Version 1.0 to 2.0

- Added changelog
- Typo fixes
- Correction of the API endpoints, to match the implementation
- Correction of diagrams 2.1, 3.1, 3.4
- Added user interfaces (section 3.6)
- Added domain assumptions (section 2.4)

### B.2 Version 2.0 to 3.0

- Added the "school of origin" dynamic, with all the related changes in the API endpoints and the database structure