

**IT 309 SOFTWARE ENGINEERING**

PROJECT DOCUMENTATION

LIBRARIAN

Prepared by:

**Dorijan Komšić**

**Kenan Lokvančić**

Proposed to:

**Nermina Durmić, Assist. Prof. Dr.**

**Aldin Kovačević, Teaching Assistant**

21.6.2023.

Contents

[1. Introduction 3](#_Toc138277794)

[1.1. About the Project 3](#_Toc138277795)

[1.2. Project Functionalities and Screenshots 3](#_Toc138277796)

[2. Project Structure 9](#_Toc138277797)

[2.1. Technologies 9](#_Toc138277798)

[2.2. Database Entities 9](#_Toc138277799)

[2.3. Design Patterns 9](#_Toc138277800)

[2.4. Tests 9](#_Toc138277801)

[3. Conclusion 10](#_Toc138277802)

# 1. Introduction

This project is a book store management system using the LAMP stack. The main idea of our application is for bookstores and such to have their own management system. It can become frustrating for owners of forementioned businesses to write everything down on paper and having to manage them in a physical format. We offer a concept of a responsive, manageable and viewable application that offers its consumers a great experience, without having to worry about their data being lost.

# 1.1. About the Project

Our application offers a responsive and intuitive user interface, ensuring a seamless experience for bookstore owners and staff. This project allows bookstore owners to make informed decisions regarding stock replenishment, pricing, and overall business operations.

**A link for the deployed project can be found on our GitHub repository:**  
<https://github.com/DorijanKom/WebProgramming2022>

# 1.2. Project Functionalities and Screenshots

The features included in the release:

**Books and Writers**

* GET /baw endpoint, returns all baws (Books and Writers) from the api
* GET /baw/@id endpoint, returns a single baw by id
* POST/baw endpoint, adds a new baw to database
* PUT /baw/@id endpoint, update a baw from the db by id

**Books**

* GET /books endpoint, returns all books from the api
* GET /books/@id endpoint, returns a single book by id
* GET /books/search/@name endpoint, returns a number of books that match the given paramater
* GET /search\_books/writer endpoint, return the provided books from the api
* POST /books endpoint, adds a new book to the db
* PUT /books/@id endpoint, updates the selected book by id

**Orders**

* GET /orders endpoint, returns all orders from the api
* GET /orders/@id endpoint, returns an order by id
* POST /orders endpoint, adds a new order to the db
* PUT /orders/@id endpoint, updates an order in db by id
* DELETE /orders/@id endpoint, deletes an order from db by its id Publishers
* GET /publishers endpoint, returns all publishers from the api
* GET /publishers/@id endpoint, returns a single publisher by id
* POST /publishers endpoint, adds a new publisher to the db
* PUT /publishers/@id endpoint, updates one publisher Purchases
* GET /purchases endpoint, returns all purchases from the API
* GET /purchases/@id endpoint, returns a single purchase by id
* POST /purchases endpoint, add a new purchase to db Users
* GET /users endpoint, returns all users from the API

**Purchases**

* GET /purchases endpoint, returns all purchases from the API
* GET /purchases/@id endpoint, returns a single purchase by id
* POST /purchases endpoint, add a new purchase to db

**Auth**

* GET /users/@id endpoint, returns a single user by id
* POST /login endpoint, with password and email for login Writers

**Writers**

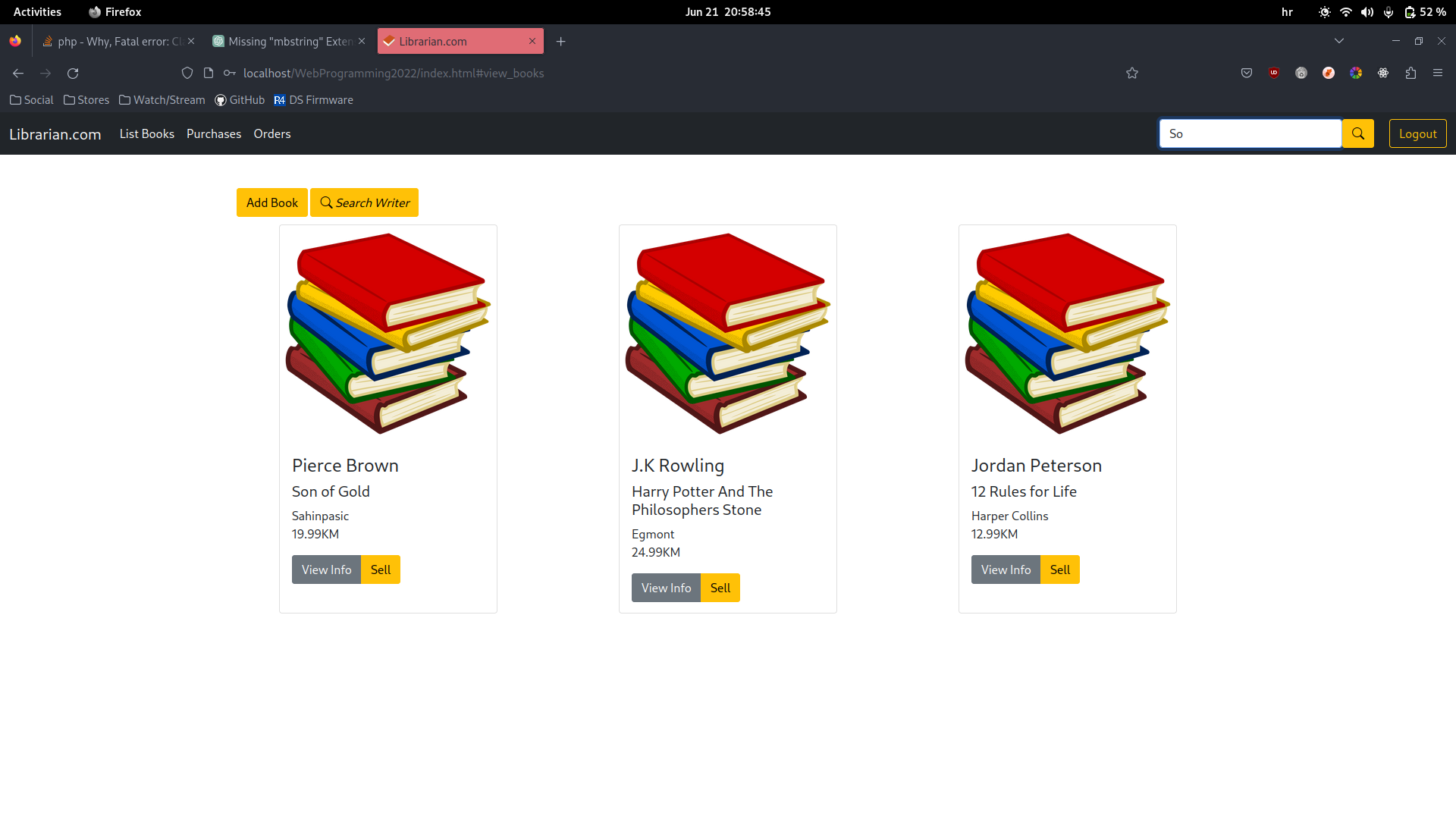
* GET /writers endpoint, returns all writers from the api
* GET /writers/@id endpoint, returns a single writer by id
* POST /writers endpoint, add a new writer to db
* PUT /writers/@id endpoint, updates a writer in db by id

**Documentation**

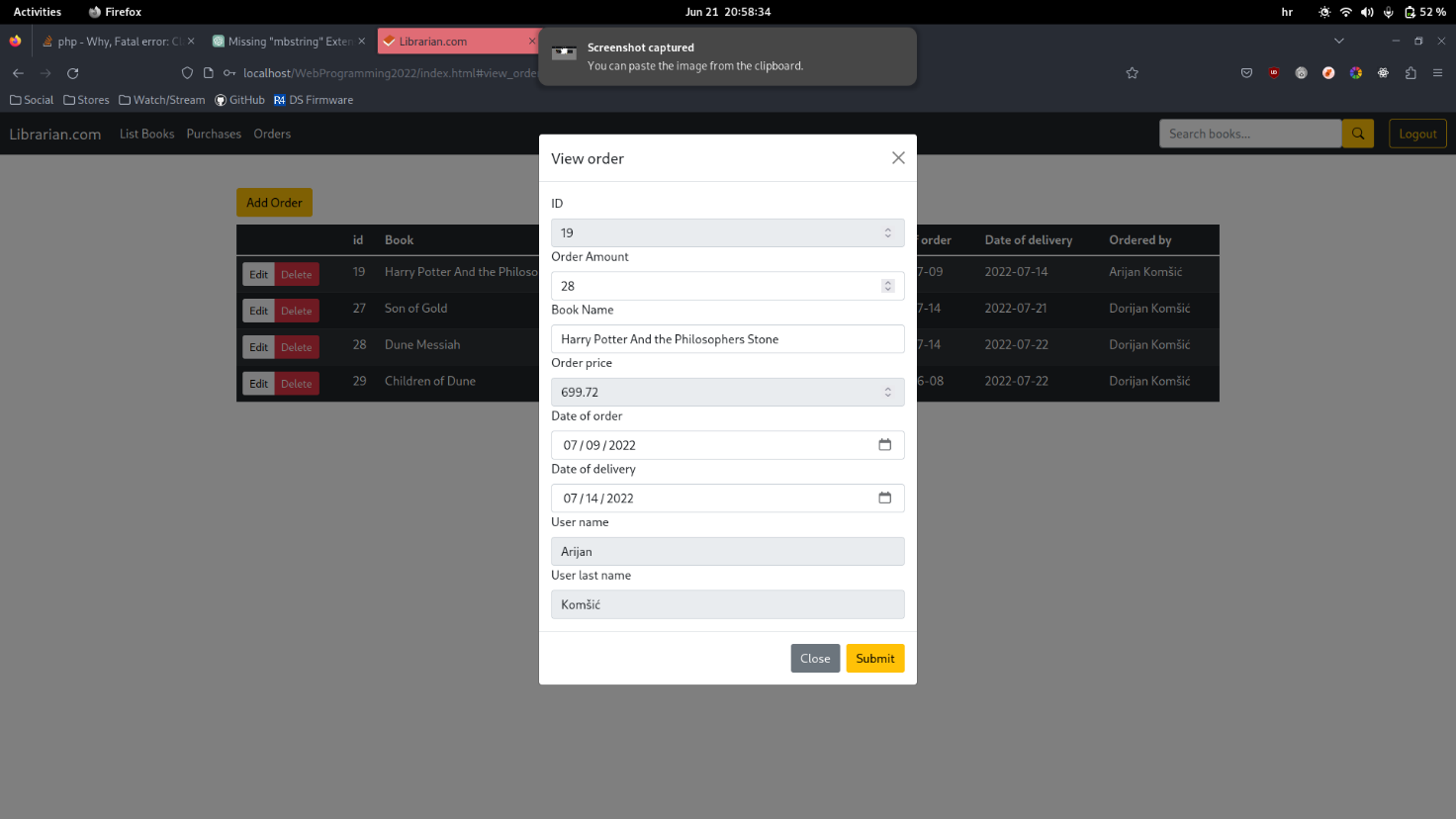
- /rest/docs endpoint, access swagger documentation

***For trying out the app, use these credentials email: dorijan.komsic@stu.ibu.edu.ba password: 12345***

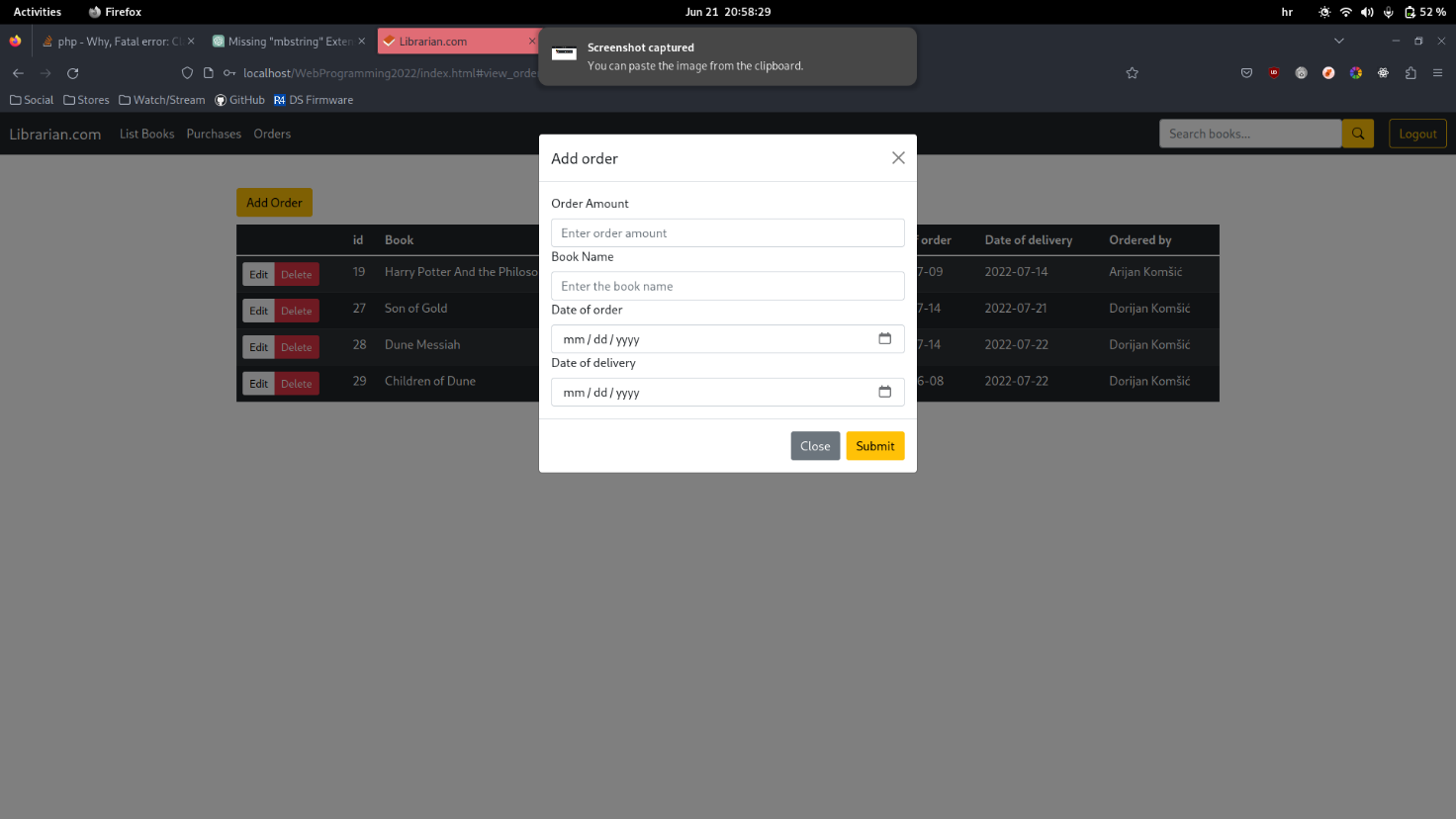
**Screenshots of the project:**



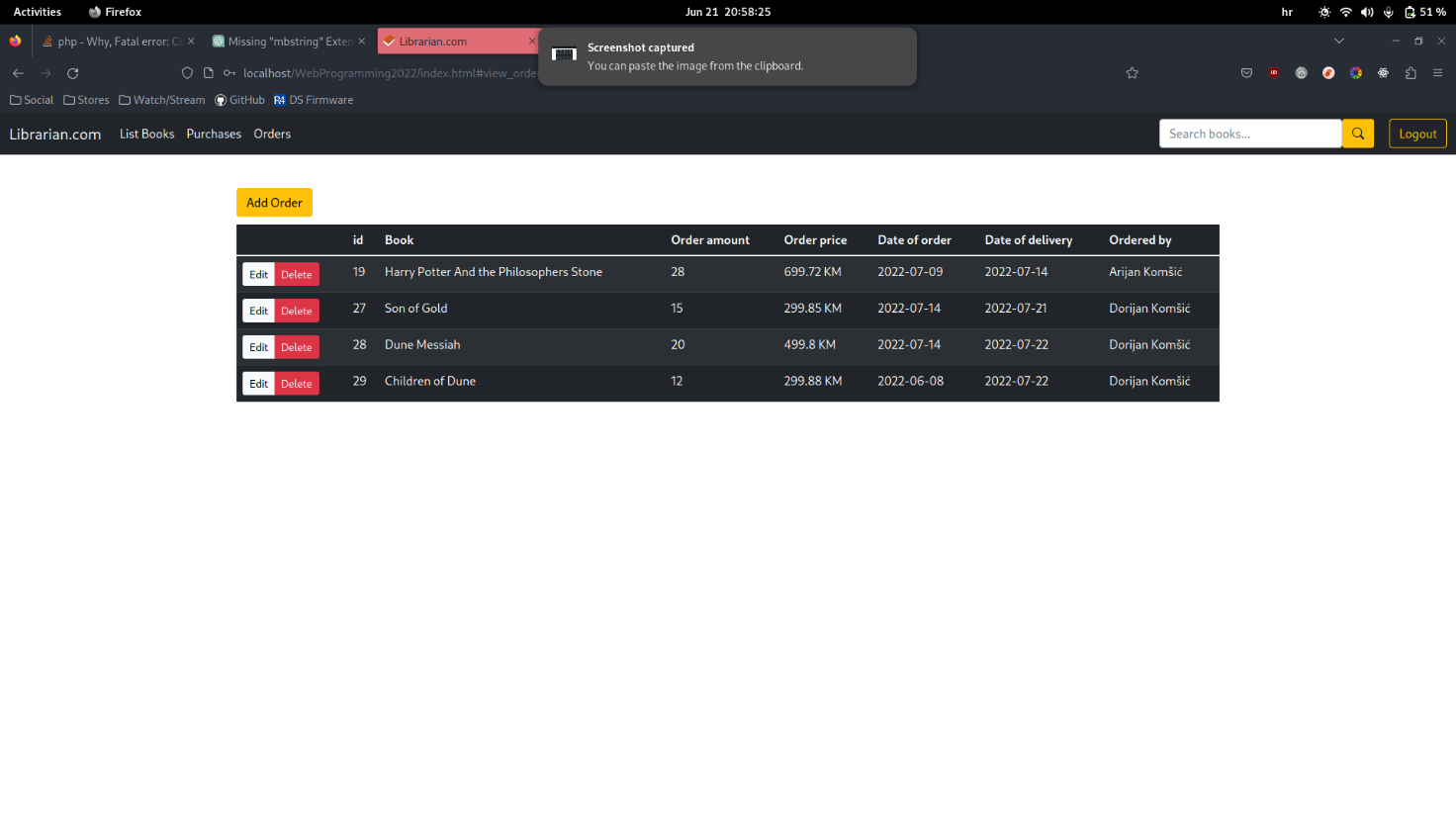
**Picture 1.** The main page



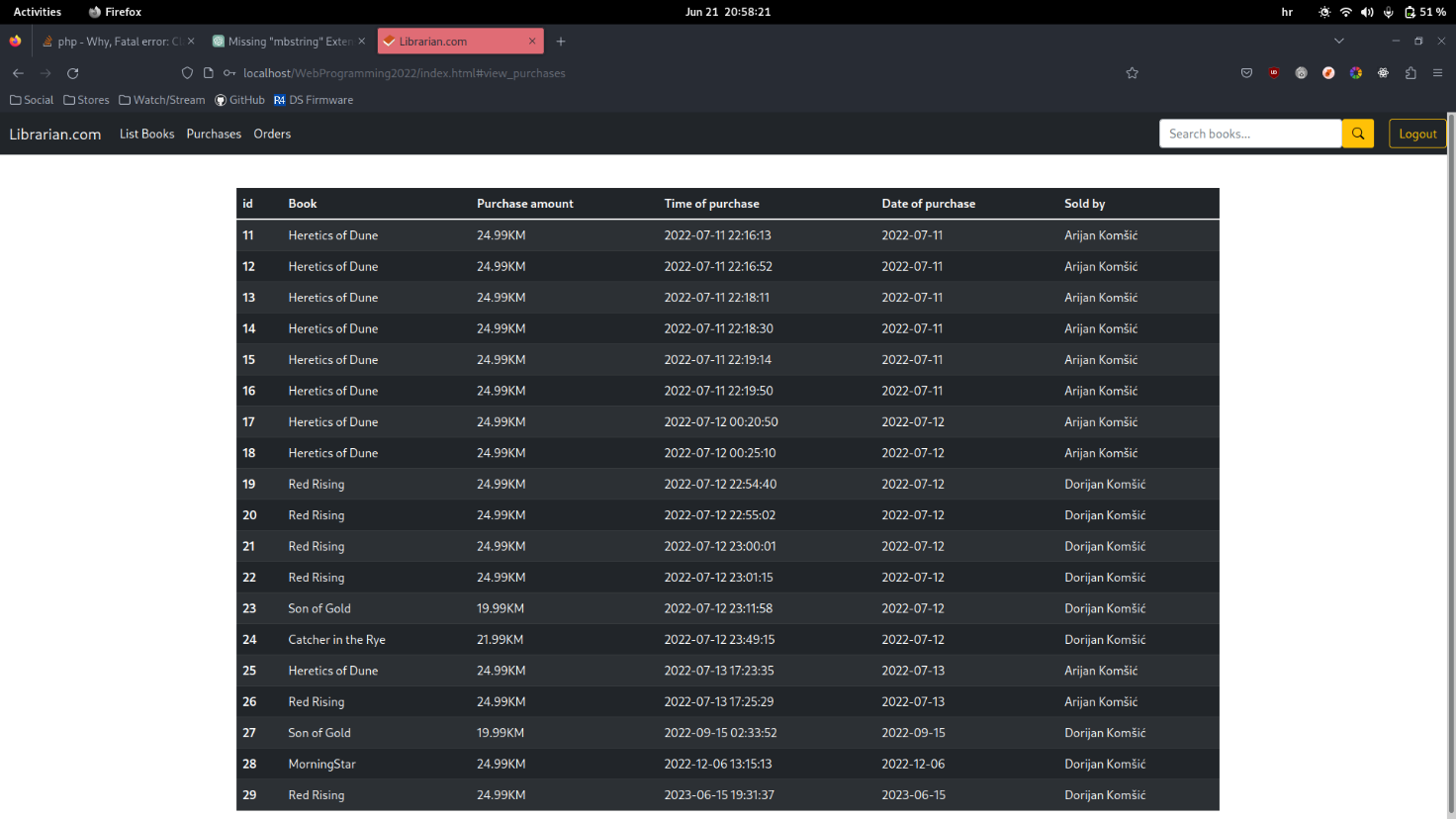
**Picture 2.** Viewing the order



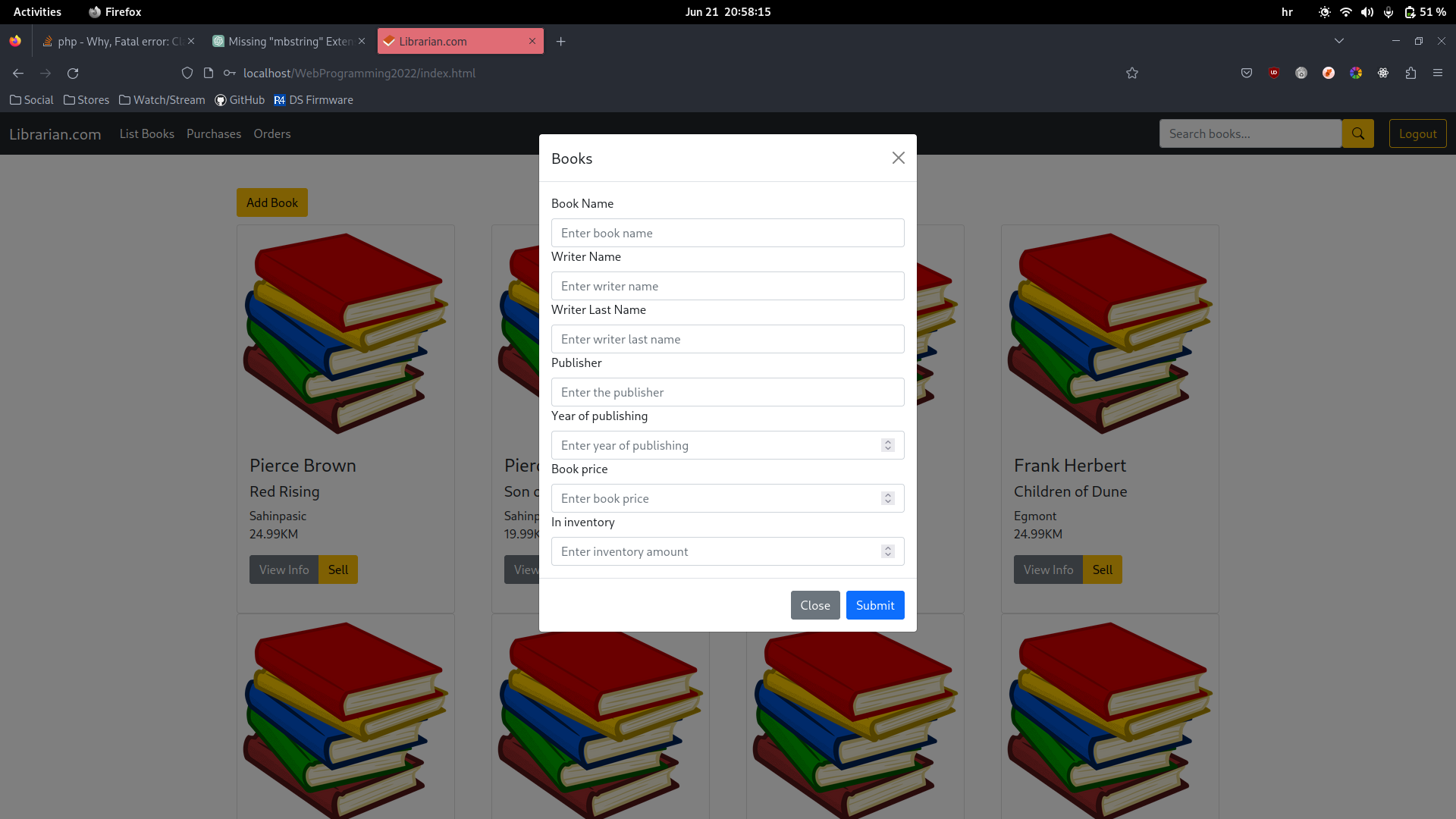
**Picture 3.** Adding an order



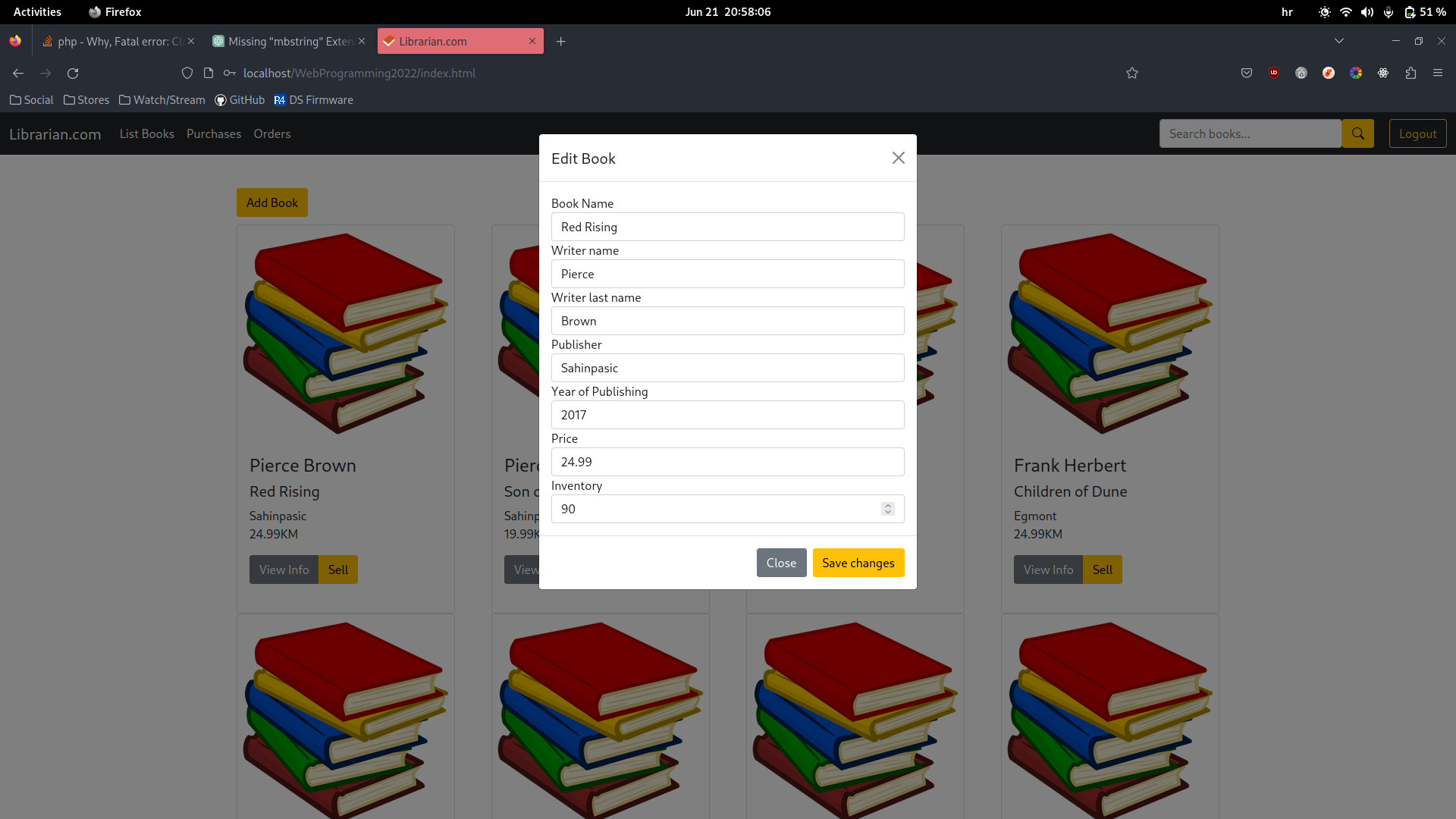
**Picture 4.** Showcase of the page with the previously added book



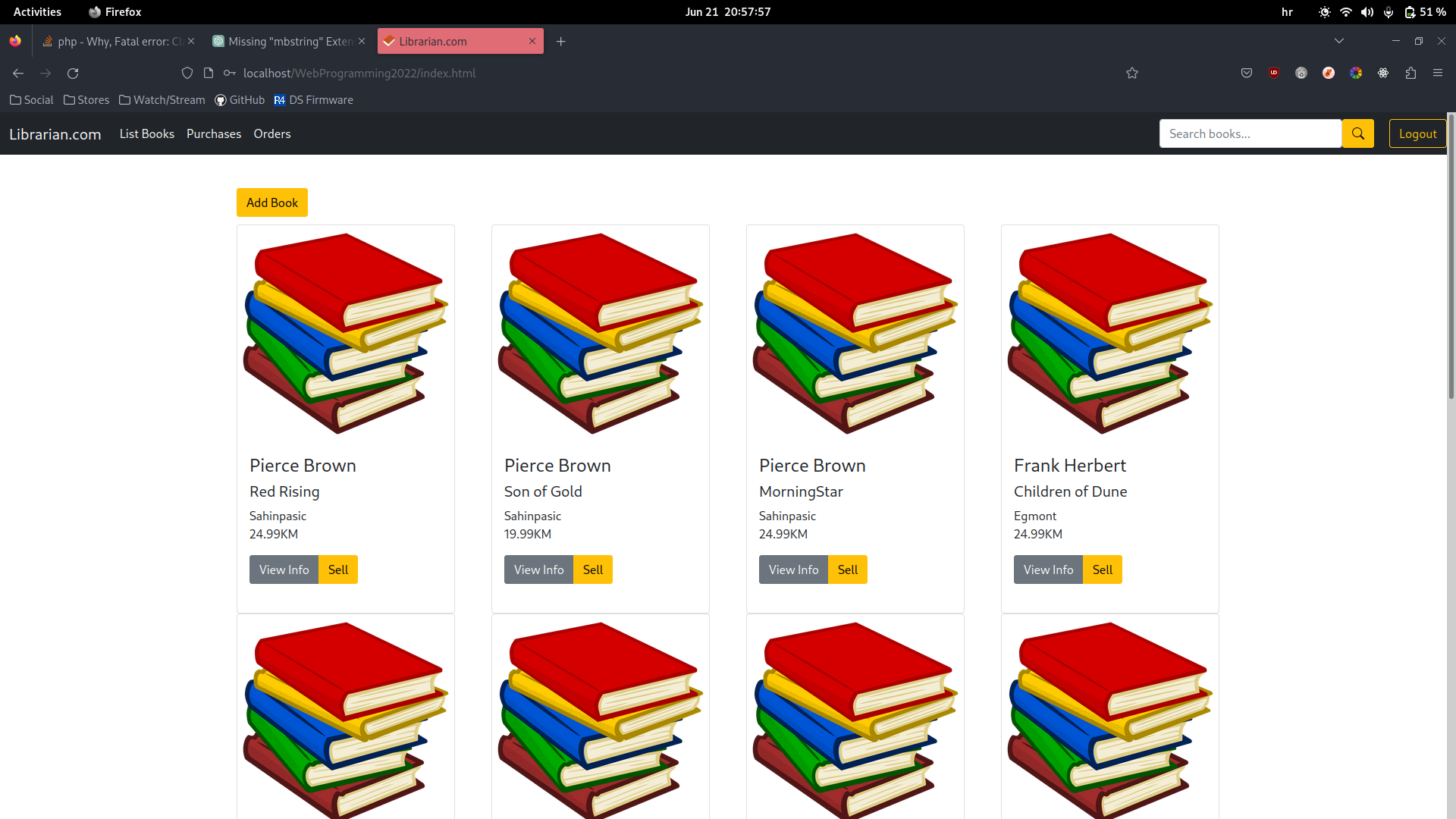
**Picture 5.** Showcase of the Purchases tab



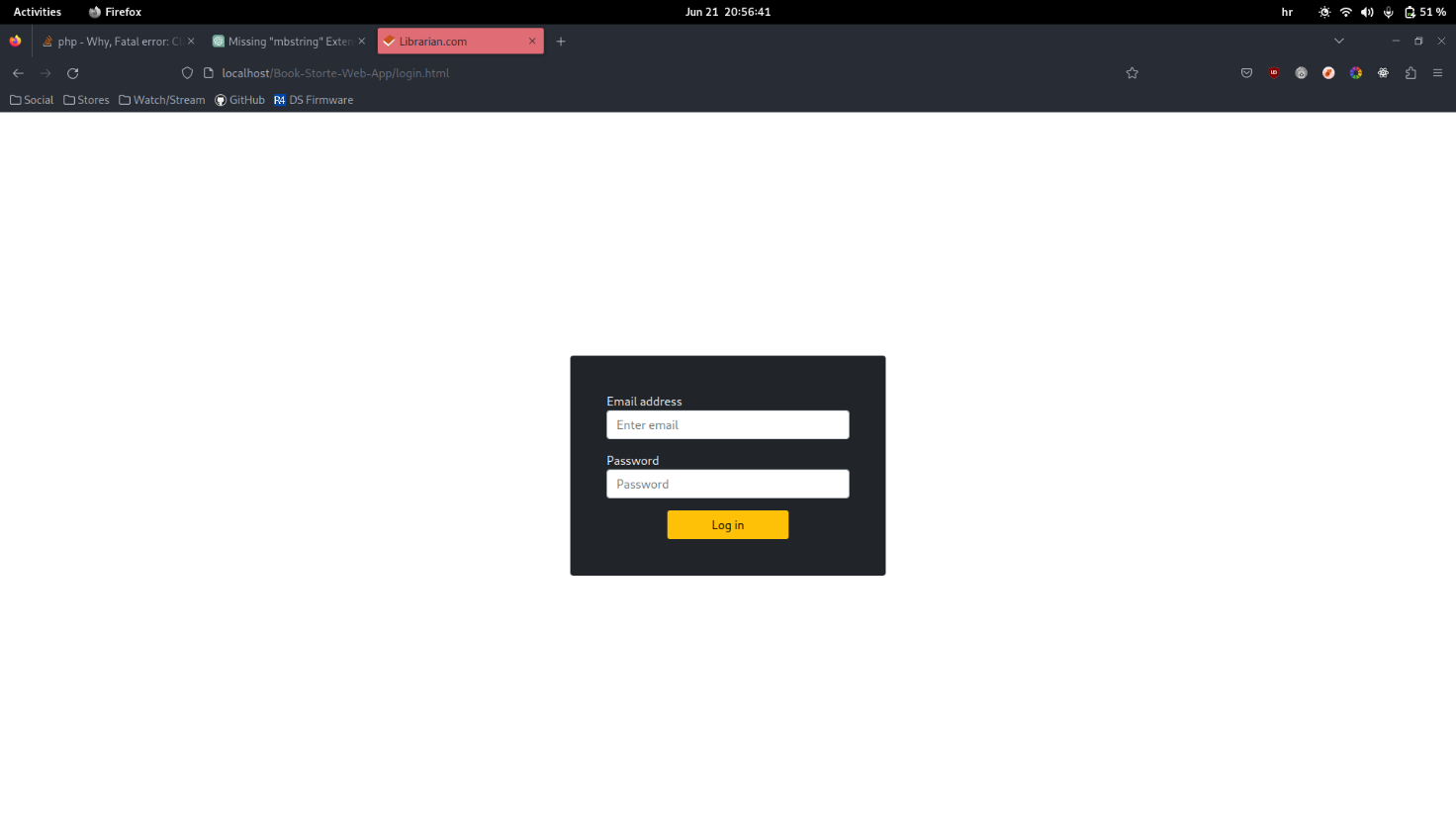
**Picture 6.** Adding a book



**Picture 7.** Editing/Updating a book



**Picture 8.** Book tab showcase



**Picture 9.** Login page

# 2. Project Structure

# 2.1. Technologies

The project was made using the LAMP stack while also using the jQuery library and FlightPHP framework.

The coding standard that we used: PSR-12

# 2.2. Database Entities

List of tables that we have in our database:

* Users
* Orders
* Purchases
* Books
* BooksAndWriters
* Writers
* Publishers

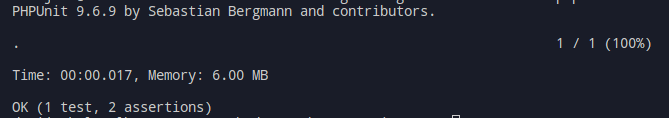
# 2.3. Design Patterns

We used MVC as our design architecture. The design patterns that we used are:

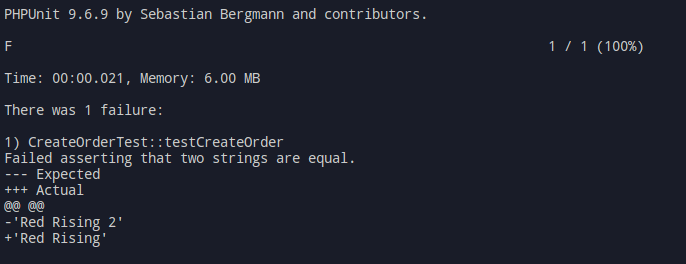
* Singleton: used in rest/DAO and they instantiate in rest/Services classes
* Front controller: Used in index.php, located in rest/index.php

# 2.4. Tests

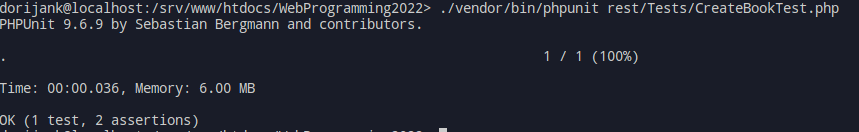
We wrote PHPUnit tests, and they are located in the rest/Test directory

Results of the tests:  


**Picture 10.** Delete Order Test



**Picture 11.** Create Order Test



**Picture 12.** Create Book Test

# 3. Conclusion

As a result, we are overall happy with how the management project for the book store has been carried out. As a team, we were successful in creating a strong and easy-to-use application that caters to the unique requirements of bookstores and streamlines their inventory management procedure. The project's main goal of providing a viewable, manageable, and responsive application that improves the overall experience for bookstore owners has been accomplished.