WEB BOOK STORE

Cristian Andres Gamez Nuñez
Catherine Melisa Maldonado Melenge

Software Modeling Foundations

BUSINESS MODEL

This business model is characterized by offering users (buyers) a wide range of products (books) that they can browse, search and buy efficiently. The purchasing process is carried out through a shopping cart, where the customer can manage the books they wish to purchase and proceed to place an order. In parallel, the application has an administrative functionality that allows sellers or administrators to manage the book catalog.

USER STORIES

01

As a User, I want to log in to my account, so i access the application's features.

02

As a customer, I want to see a list of available books, so I can choose which one to buy.

03

As a customer, I want to search for a book by some characteristic (author, ISBN, etc.), to quickly find the one I'm looking for.

04

As a customer, I want to be able to see the details of a book, to know its description, price, and author before deciding whether to buy it.

05

As a customer, I want to be able to add books to a shopping cart, so I can buy them all at once.

06

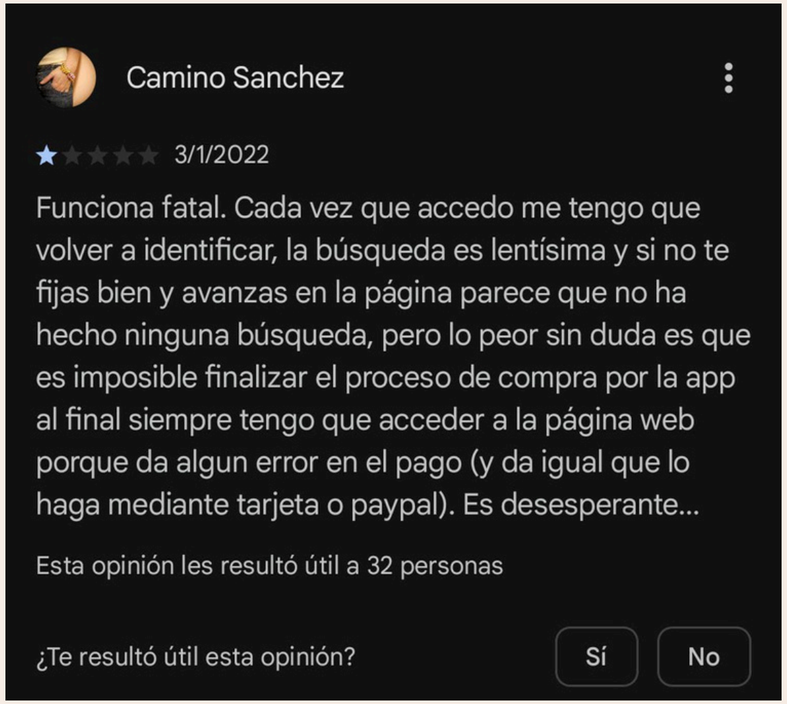
As a customer, I want to be able to remove books from the shopping cart, to adjust my order before checkout.

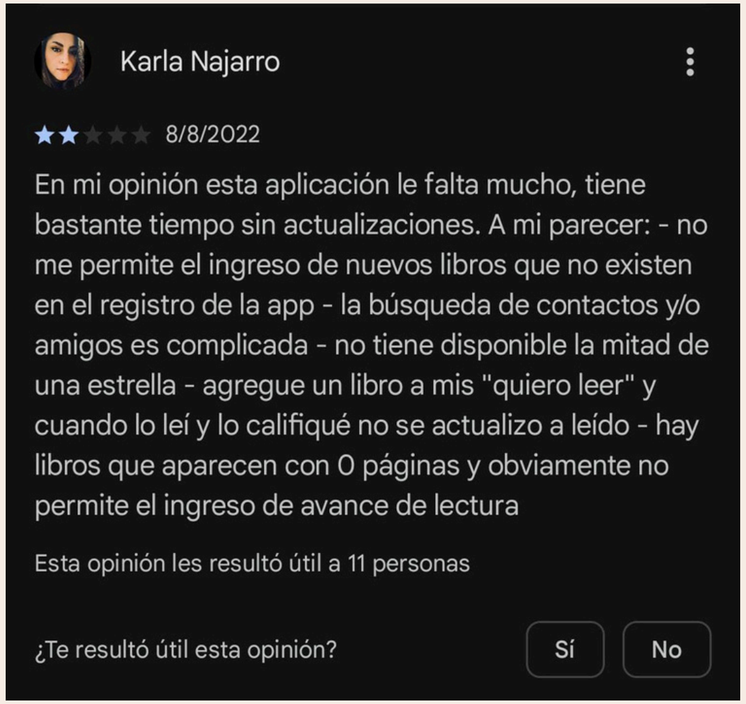
07

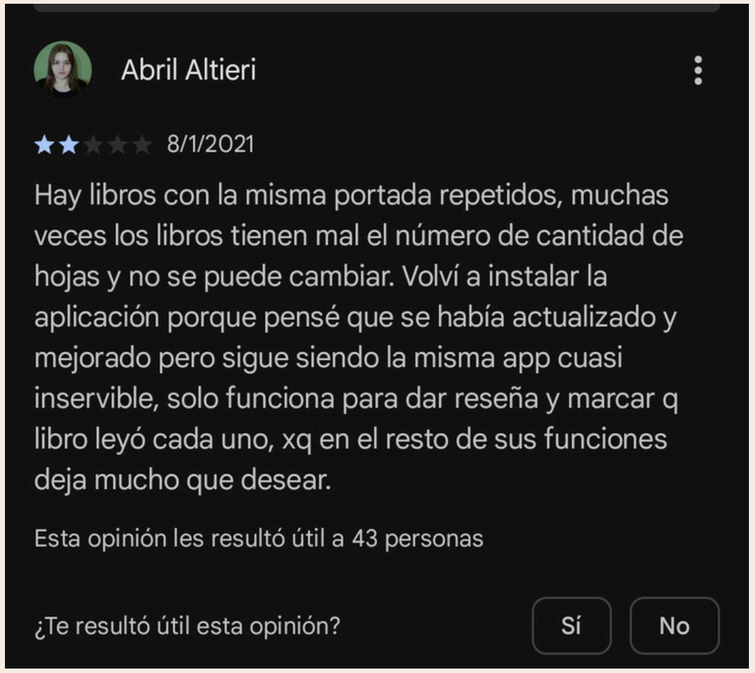
As an administrator, I want to be able to add new books to the catalog, to keep the store up to date.

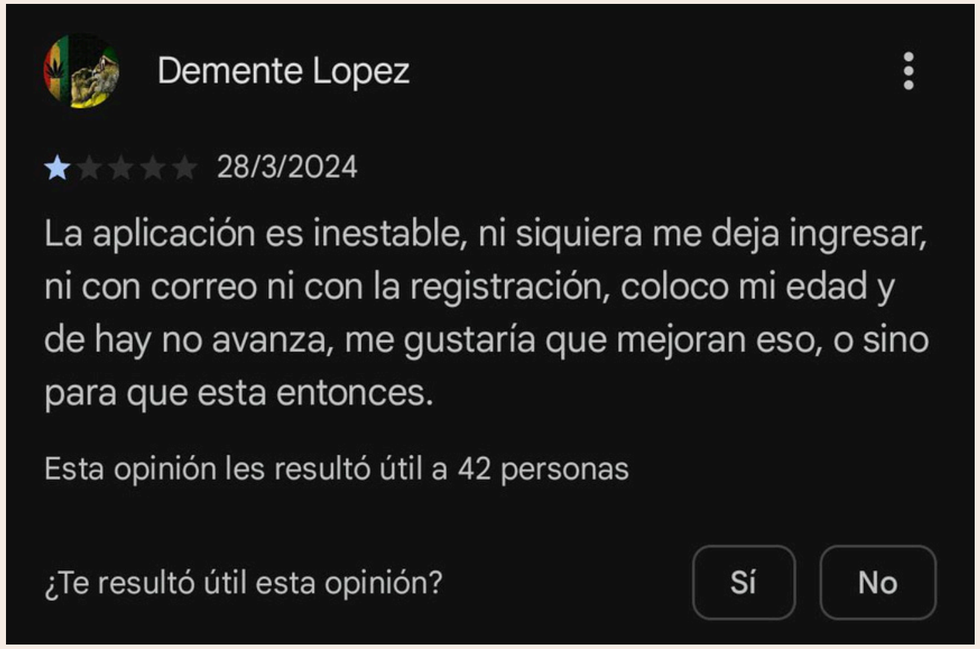
As an administrator, I want to be able to modify the details of a book (price, stock, description), to adjust the product information.

08


Camino Sanchez
3/1/2022
Funciona fatal. Cada vez que accedo me tengo que volver a identificar, la búsqueda es lentísima y si no te fijas bien y avanzas en la página parece que no ha hecho ninguna búsqueda, pero lo peor sin duda es que es imposible finalizar el proceso de compra por la app al final siempre tengo que acceder a la página web porque da algun error en el pago (y da igual que lo haga mediante tarjeta o paypal). Es desesperante...
Esta opinión les resultó útil a 32 personas
¿Te resultó útil esta opinión?
Sí
No


Karla Najarro
8/8/2022
En mi opinión esta aplicación le falta mucho, tiene bastante tiempo sin actualizaciones. A mi parecer: - no me permite el ingreso de nuevos libros que no existen en el registro de la app - la búsqueda de contactos y/o amigos es complicada - no tiene disponible la mitad de una estrella - agregue un libro a mis "quiero leer" y cuando lo leí y lo califiqué no se actualizo a leído - hay libros que aparecen con 0 páginas y obviamente no permite el ingreso de avance de lectura
Esta opinión les resultó útil a 11 personas
¿Te resultó útil esta opinión?
Sí
No


Abril Altieri
8/1/2021
Hay libros con la misma portada repetidos, muchas veces los libros tienen mal el número de cantidad de hojas y no se puede cambiar. Volví a instalar la aplicación porque pensé que se había actualizado y mejorado pero sigue siendo la misma app cuasi inservible, solo funciona para dar reseña y marcar q libro leyó cada uno, xq en el resto de sus funciones deja mucho que desear.
Esta opinión les resultó útil a 43 personas
¿Te resultó útil esta opinión?
Sí
No


Demente Lopez
28/3/2024
La aplicación es inestable, ni siquiera me deja ingresar, ni con correo ni con la registración, coloco mi edad y de hay no avanza, me gustaría que mejoran eso, o sino para que esta entonces.
Esta opinión les resultó útil a 42 personas
¿Te resultó útil esta opinión?
Sí
No

FUNCTIONAL REQUIREMENTS:

01

Book Management:

- Data Loading: The system must allow loading book data from a file.
- Book Listing: The system must allow listing all the books available in the catalog.
- Book Search: The system must allow searching for books by different criteria (author, ISBN, title).

02

Shopping Cart:

- Cart Creation: The system must allow creating a shopping cart for a user.
- Add Items: The system must allow adding items to the shopping cart.
- Delete Items: The system must allow removing items from the shopping cart.
- Show Cart: The system must allow displaying the contents of the shopping cart

03

Catalog:

- Catalog Initialization: The system must allow initializing the book catalog.
- Add Books to Catalog: The system must allow adding books to the catalog.
- Catalog Search: The system must allow searching for books in the catalog using different search strategies.

NON-FUNCTIONAL REQUIREMENTS:

01

Performance:

- The system should be able to handle a large number of books and users without degrading performance.
- Searches should be fast and efficient.

02

Maintainability:

- The code should be modular and follow design principles that make it easy to maintain and extend.
- Documentation should be clear and complete.

03

Portability:

- The system should be portable and run in different environments without problems. Docker is used to ensure portability.
- The system should be scalable to handle an increase in the number of users and books.

TECHNICAL CONSIDERATIONS:

01

Architecture:

- Modularity: The system is divided into clear modules (repositories, services, controllers), which facilitates maintainability and scalability.
- Design Patterns: Design patterns such as Factory Method, Strategy, and Singleton are used to solve specific problems efficiently.

02

Technologies Used:

- FastAPI: Used to create the system API, providing endpoints for different functionalities.
- Spring: Used to create the system API, providing endpoints for different functionalities.
- Docker: Used to ensure portability and consistency of the runtime environment.

03

Dependency Management:

- Poetry:It is used to manage backend dependencies in python, ensuring that all necessary libraries are installed.
- Maven: It is used to manage backend dependencies in Java, ensuring that all necessary libraries are installed.

CRC CARDS

Book	
Responsibilities	**Collaborators**
Save information about the book	Catalog
Show details about the book to the user	Shopping Cart

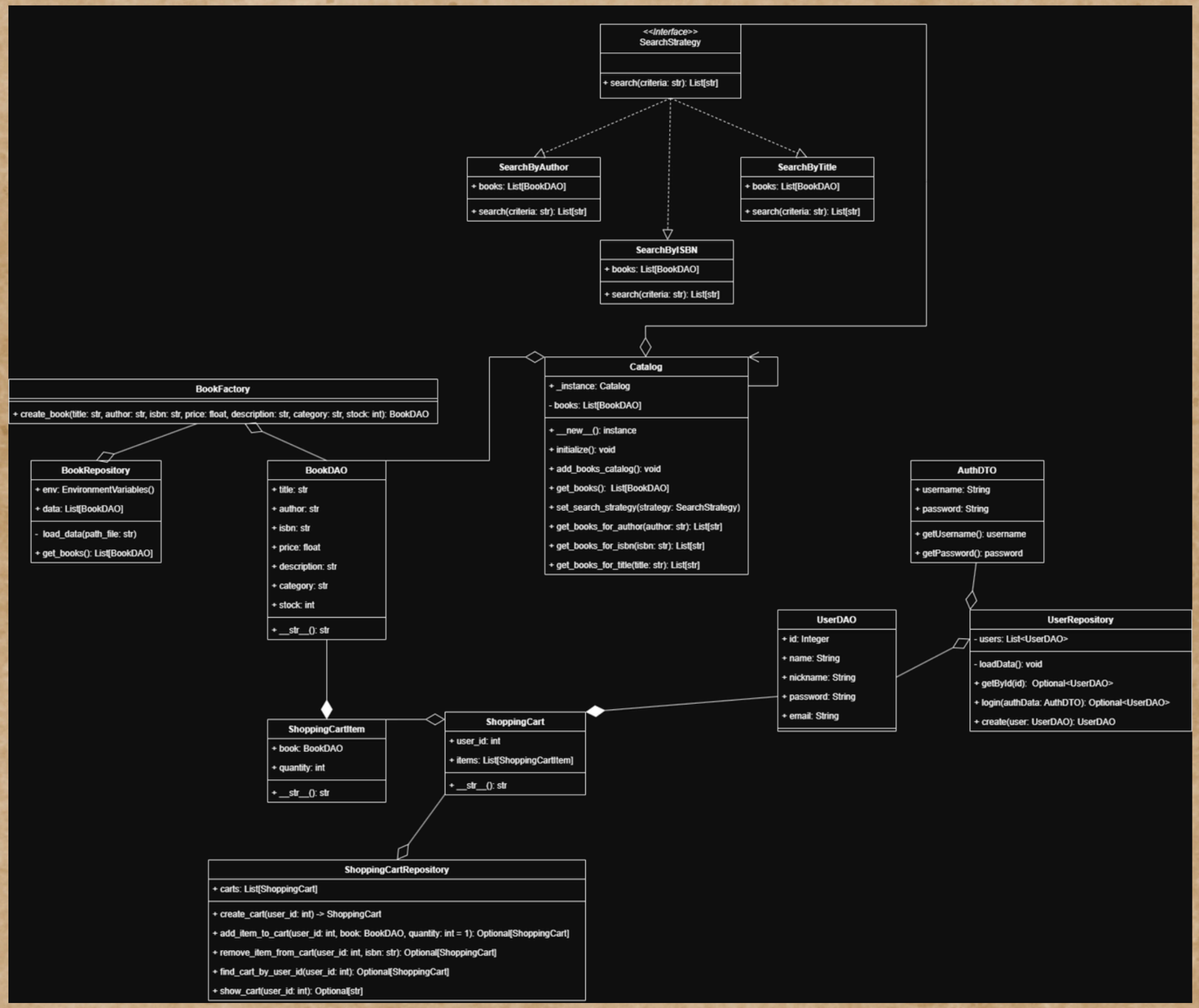
Buyer	
Responsibilities	**Collaborators**
View the book catalog	Shopping Cart
Select books and add them to the shopping cart	Catalog
Place orders and complete purchases	Order

Catalog	
Responsibilities	**Collaborators**
Manage book collection	Buyer
Allow book search	Book
Allow filtering by category, price, author, etc.	

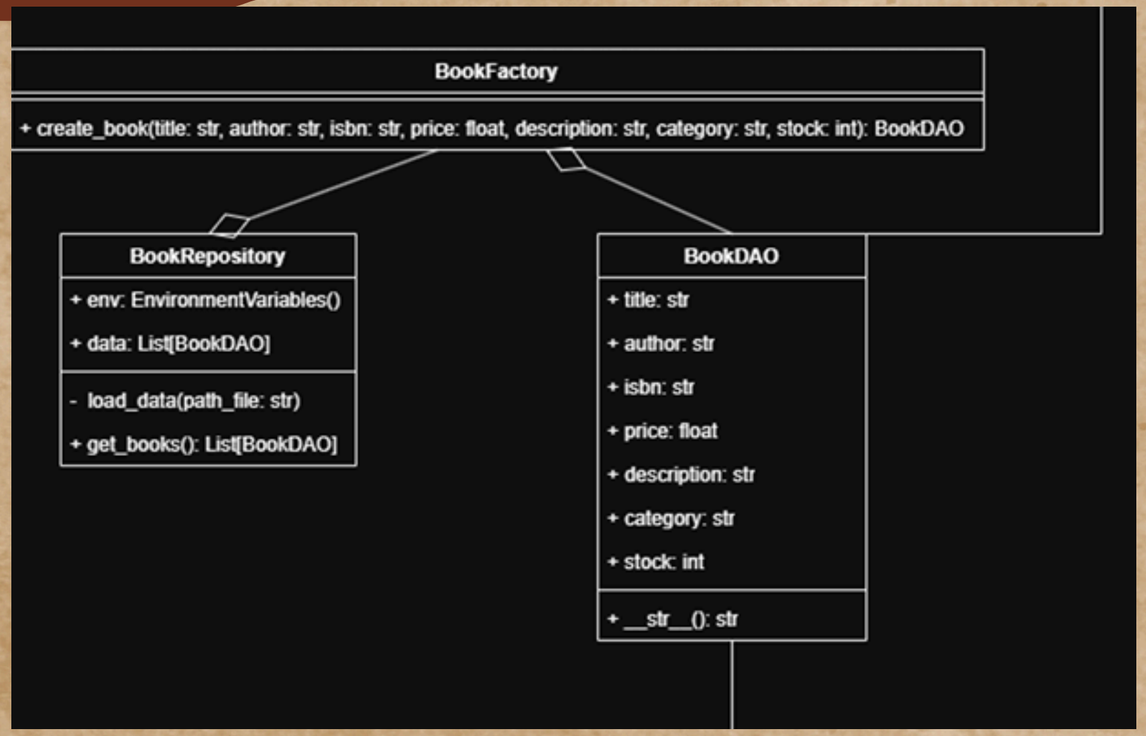
Seller	
Responsibilities	**Collaborators**
Add, modify and delete books in the catalog	Book
Manage inventory	Catalog

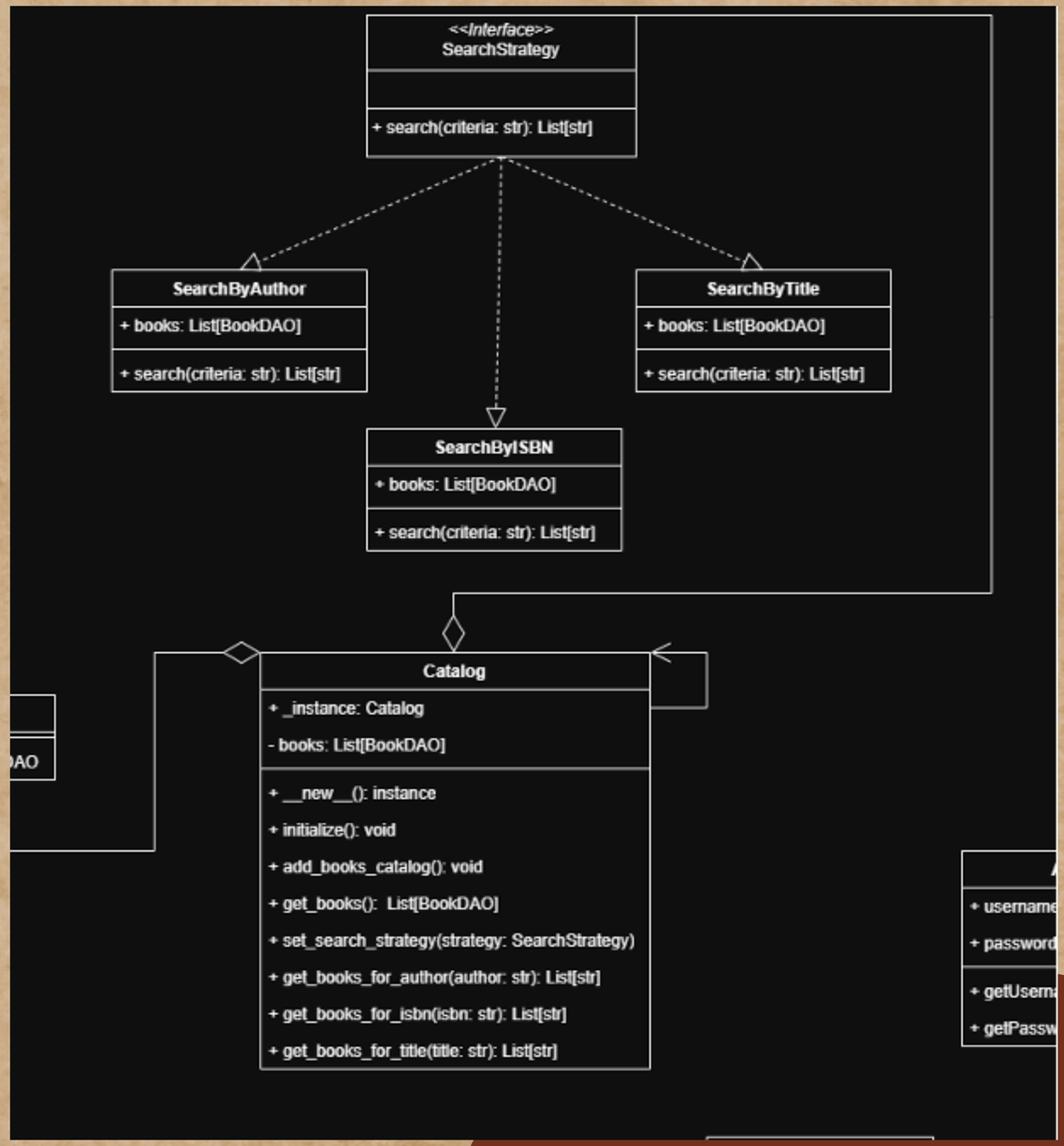
Shooping Cart	
Responsibilities	**Collaborators**
Add and remove books selected by the buyer	Book
Calculate the total of the Purchase	Buyer

CLASS DIAGRAM

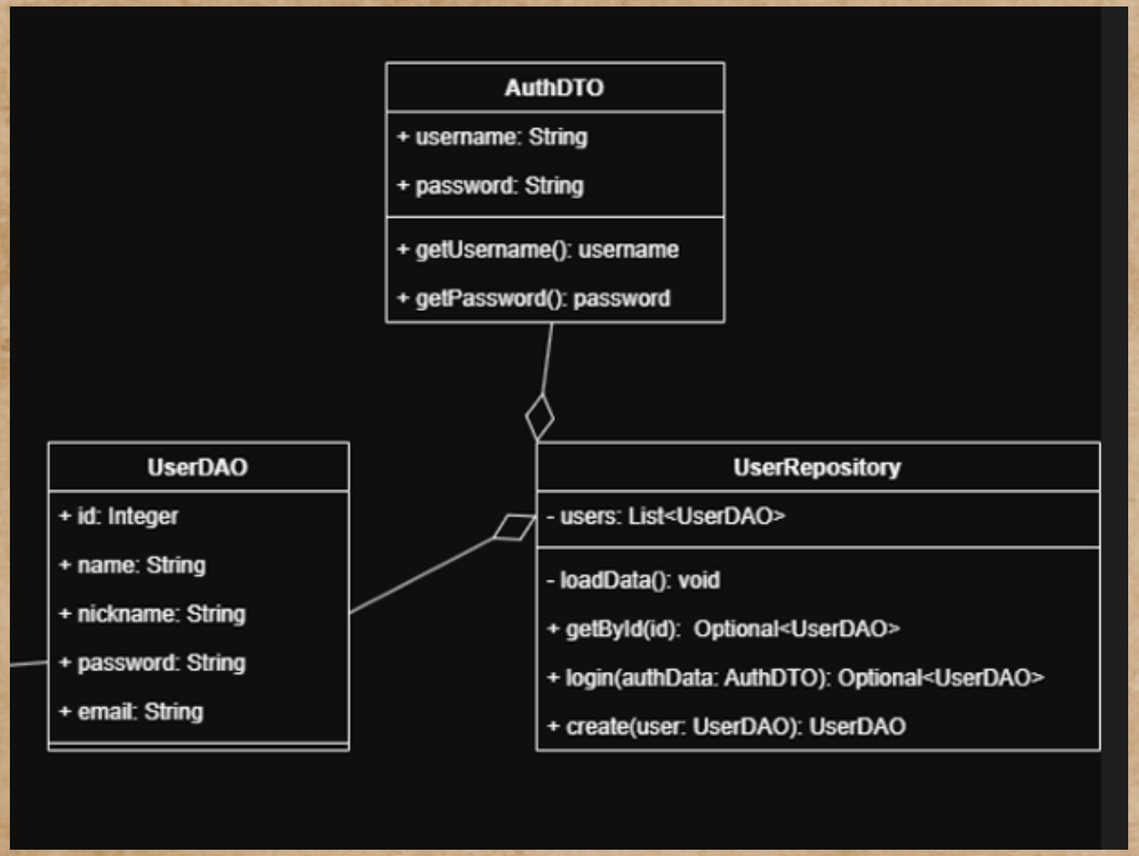


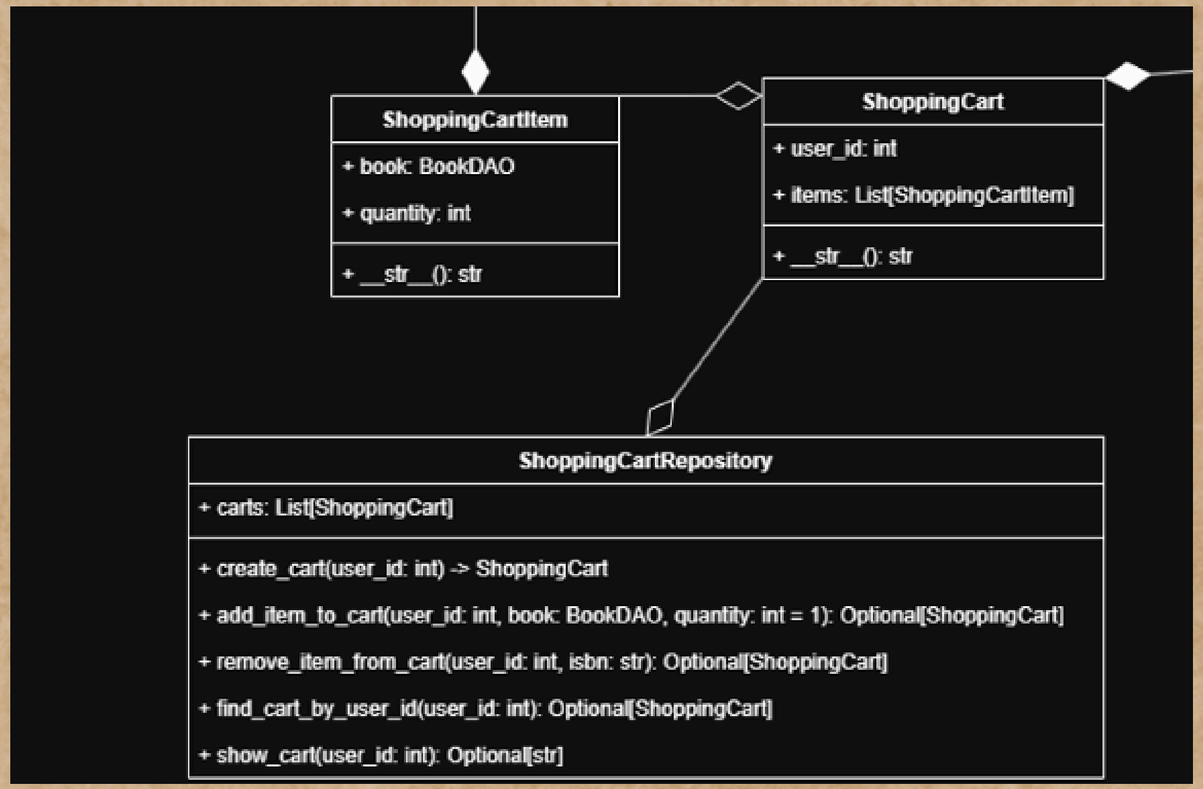
COMPONENTS


BookFactory
+ create_book(title: str, author: str, isbn: str, price: float, description: str, category: str, stock: int): BookDAO
BookRepository
+ env: EnvironmentVariables()
+ data: List[BookDAO]
- load_data(path_file: str)
+ get_books(): List[BookDAO]
BookDAO
+ title: str
+ author: str
+ isbn: str
+ price: float
+ description: str
+ category: str
+ stock: int
+ __str__(): str


<<Interface>>
SearchStrategy
+ search(criteria: str): List[str]
SearchByAuthor
+ books: List[BookDAO]
+ search(criteria: str): List[str]
SearchByTitle
+ books: List[BookDAO]
+ search(criteria: str): List[str]
SearchByISBN
+ books: List[BookDAO]
+ search(criteria: str): List[str]
Catalog
+ _instance: Catalog
- books: List[BookDAO]
+ __new__(): instance
+ initialize(): void
+ add_books_catalog(): void
+ get_books(): List[BookDAO]
+ set_search_strategy(strategy: SearchStrategy)
+ get_books_for_author(author: str): List[str]
+ get_books_for_isbn(isbn: str): List[str]
+ get_books_for_title(title: str): List[str]

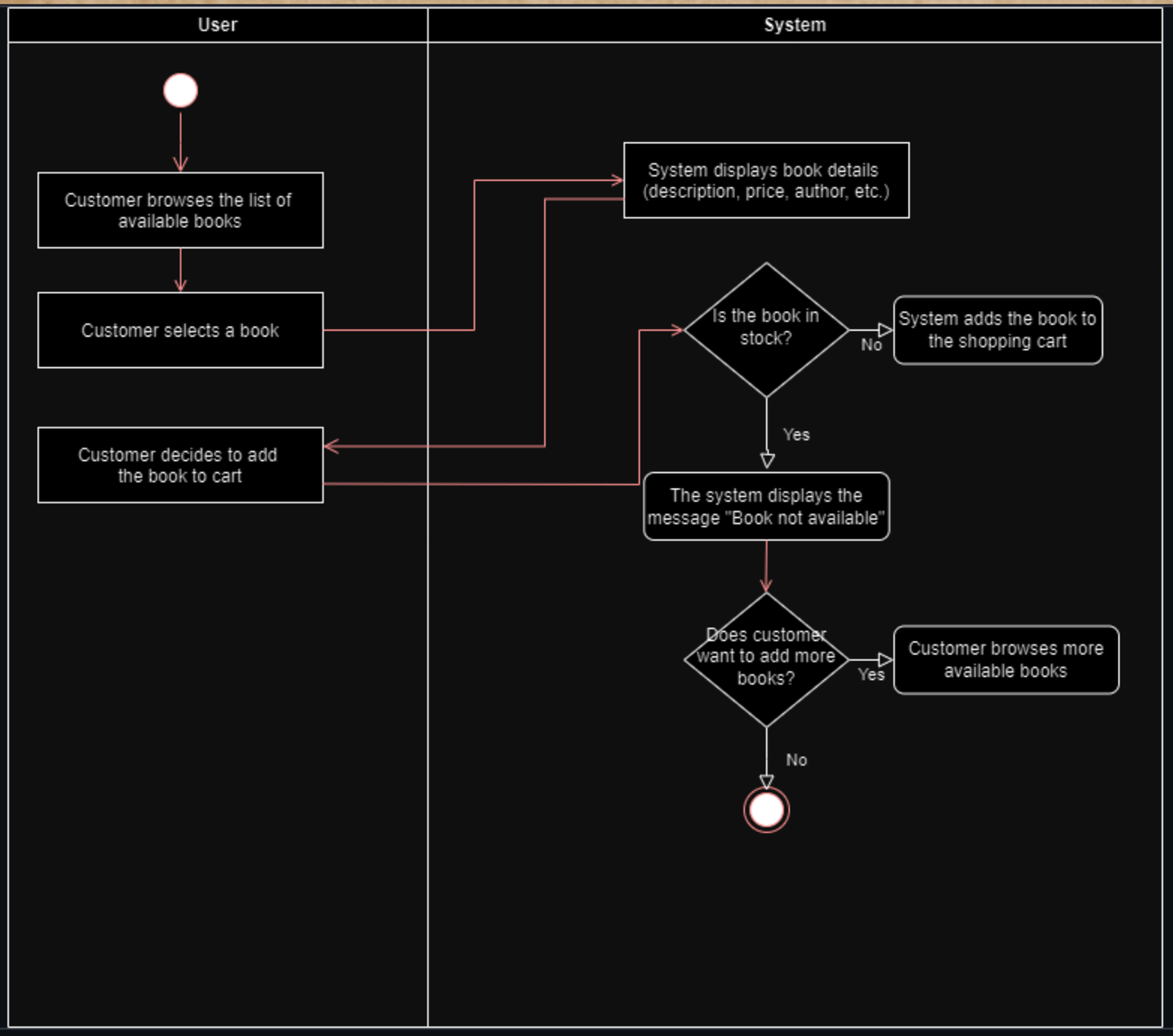
COMPONENTS


AuthDTO
+ username: String
+ password: String
+ getUsername(): username
+ getPassword(): password
UserDAO
+ id: Integer
+ name: String
+ nickname: String
+ password: String
+ email: String
UserRepository
- users: List<UserDAO>
- loadData(): void
+ getById(id): Optional<UserDAO>
+ login(authData: AuthDTO): Optional<UserDAO>
+ create(user: UserDAO): UserDAO

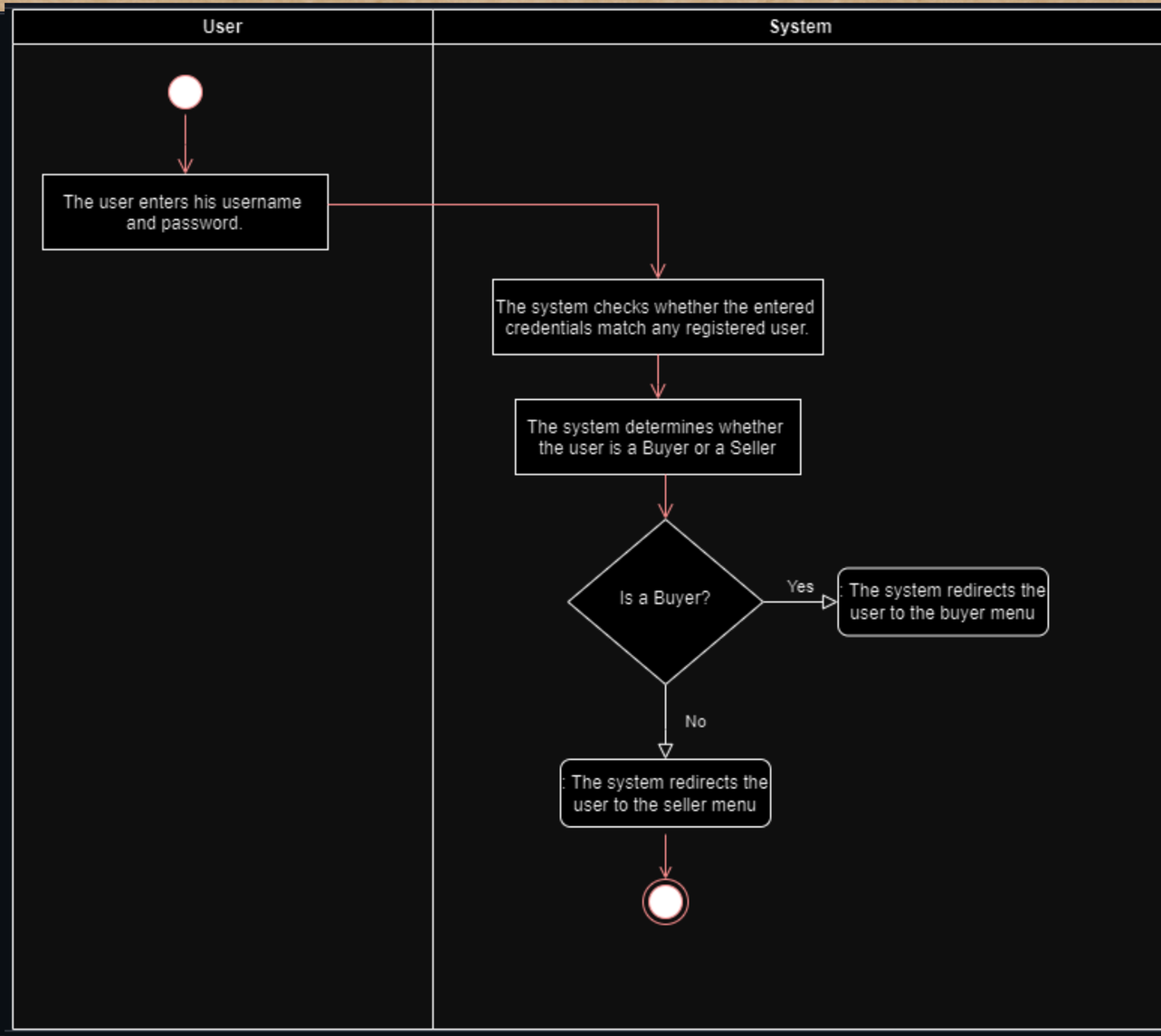

ShoppingCartItem
+ book: BookDAO
+ quantity: int
+ __str__(): str
ShoppingCart
+ user_id: int
+ items: List[ShoppingCartItem]
+ __str__(): str
ShoppingCartRepository
+ carts: List[ShoppingCart]
+ create_cart(user_id: int) -> ShoppingCart
+ add_item_to_cart(user_id: int, book: BookDAO, quantity: int = 1): Optional[ShoppingCart]
+ remove_item_from_cart(user_id: int, isbn: str): Optional[ShoppingCart]
+ find_cart_by_user_id(user_id: int): Optional[ShoppingCart]
+ show_cart(user_id: int): Optional[str]

ACTIVITIES DIAGRAMS

addBooksCart

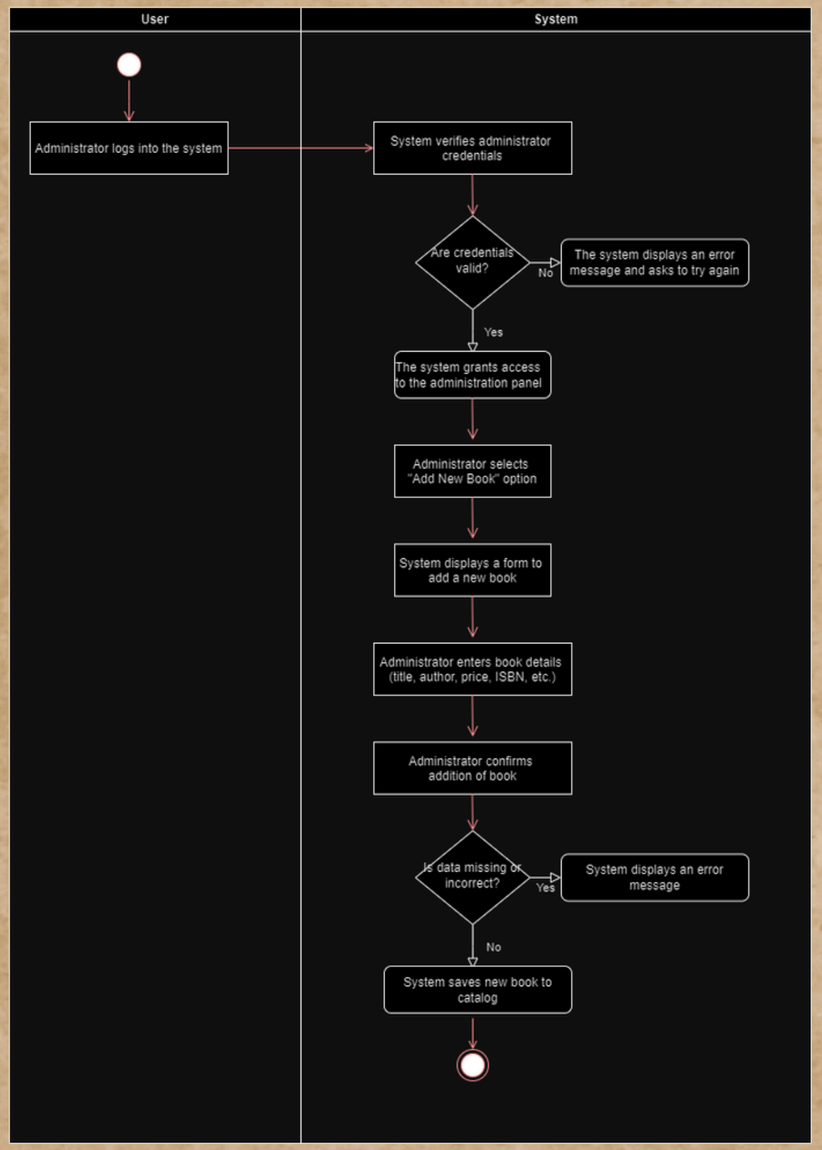


login

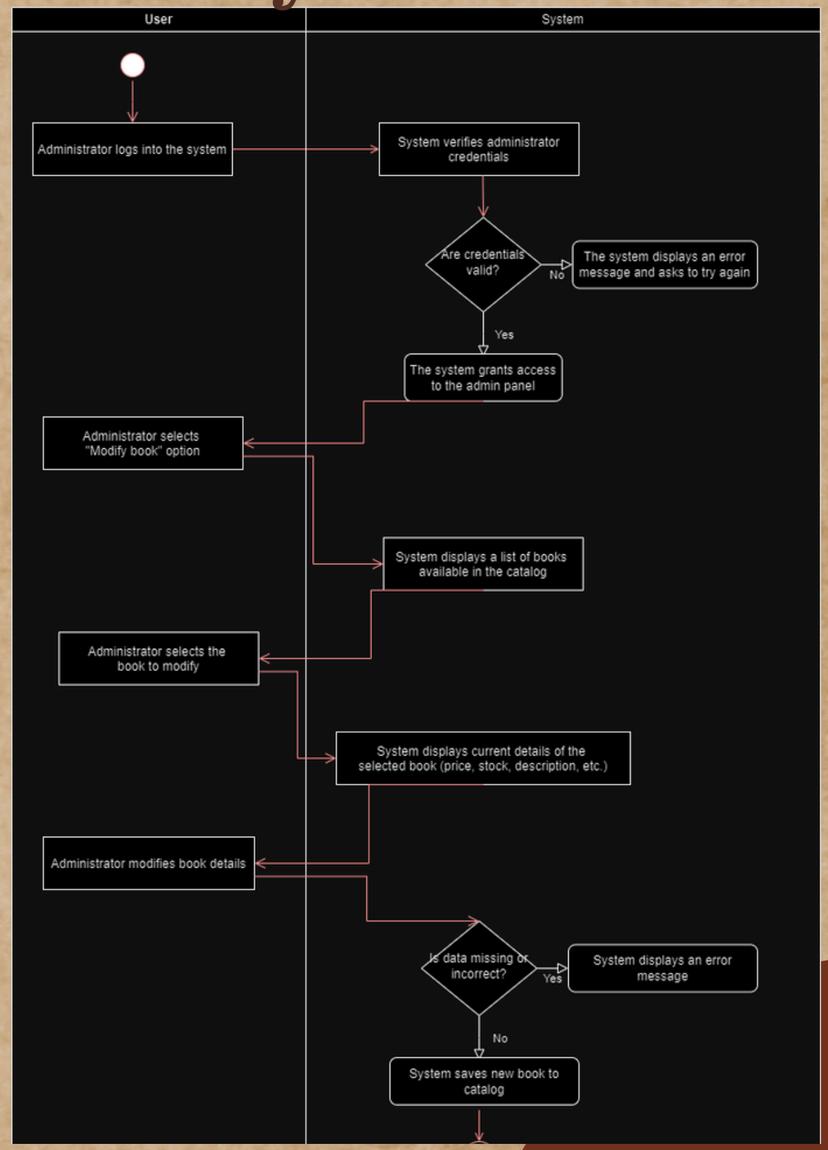


ACTIVITIES DIAGRAMS

addNewBooks

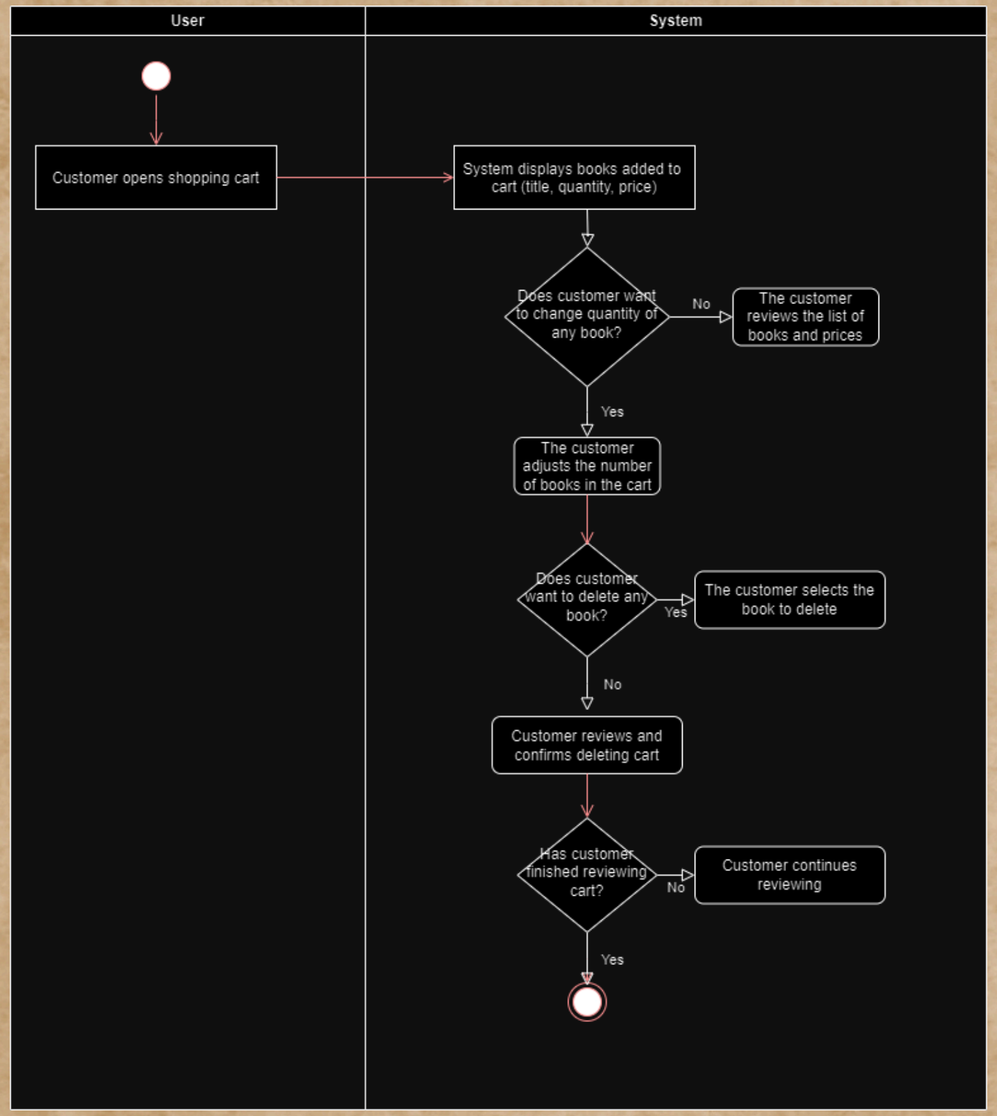


modifyDetailsBooks

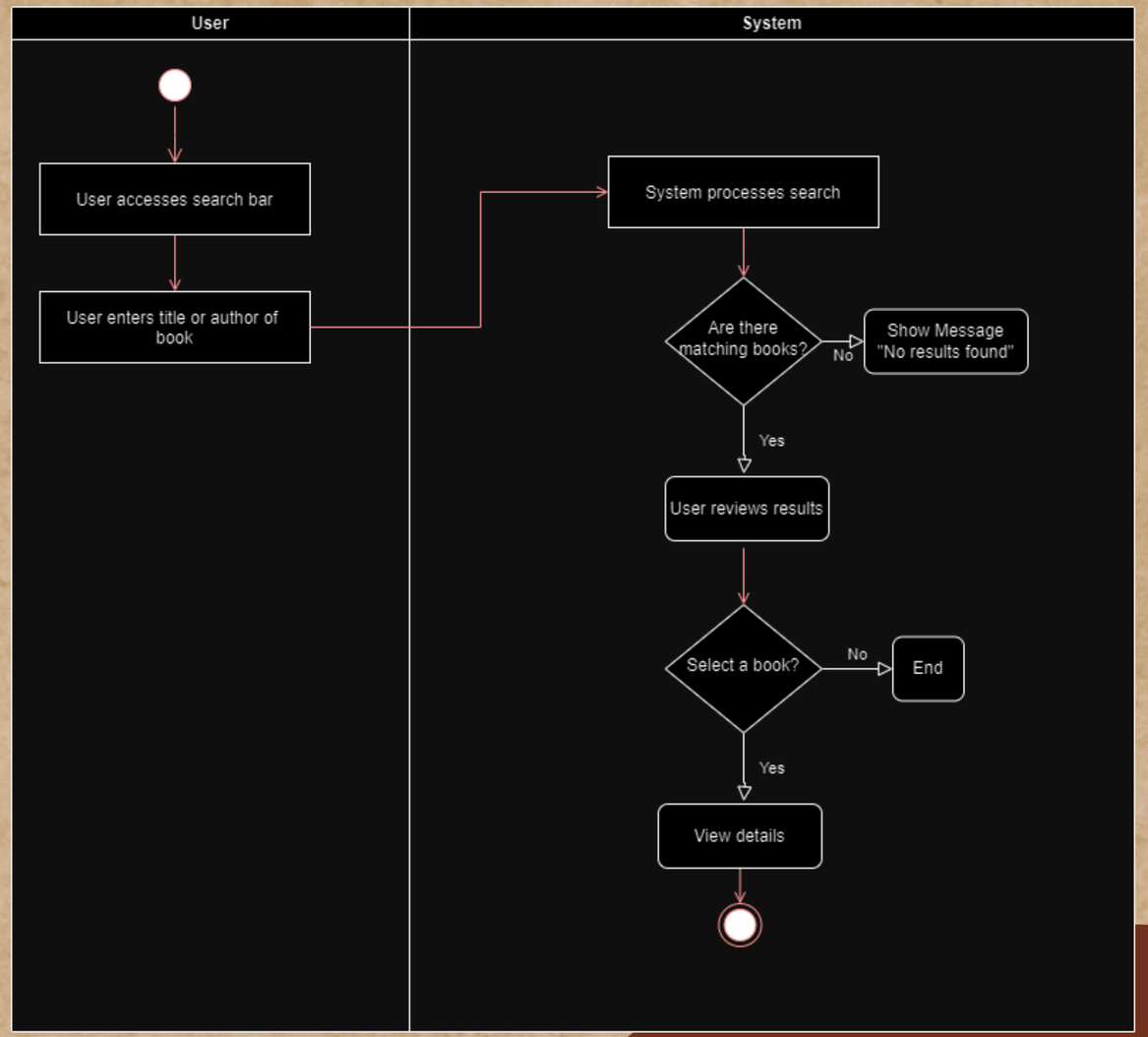


ACTIVITIES DIAGRAMS

reviewBooksCart

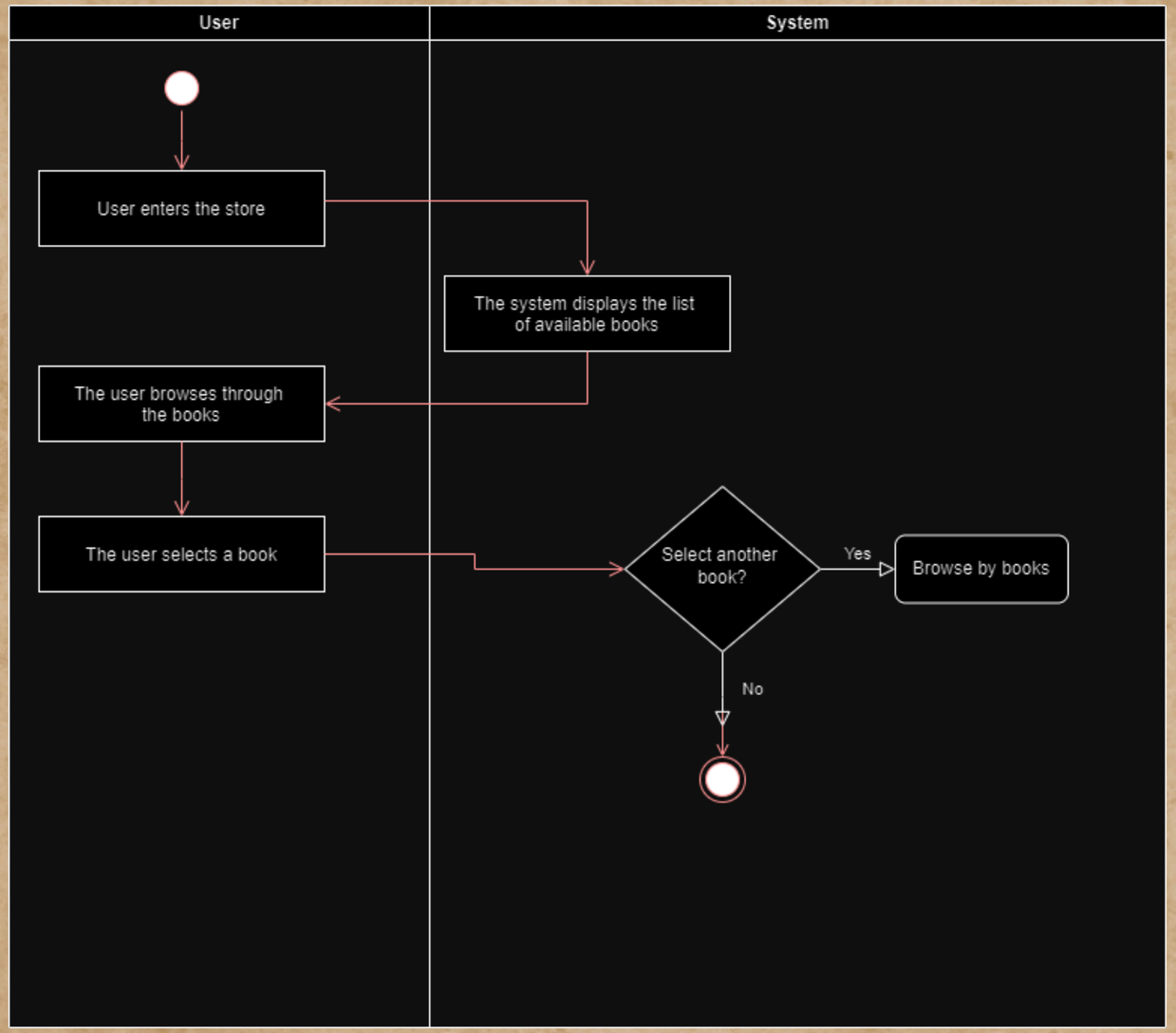


searchBooks

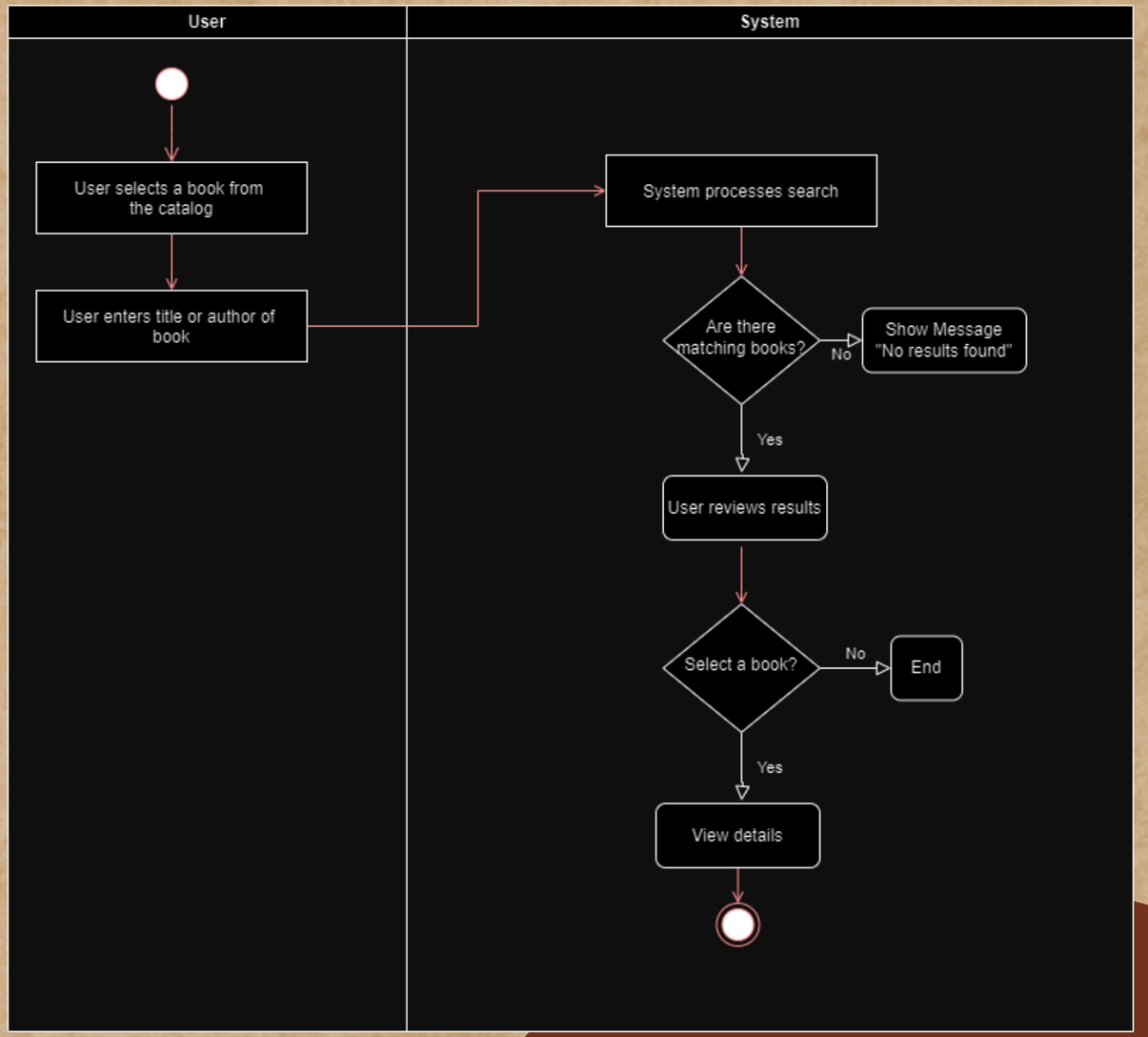


ACTIVITIES DIAGRAMS

showBooks.



showDetailsBooks



GOOD PRACTICES

SOLID PRINCIPLES

The architecture follows SOLID principles, which improves the flexibility and maintainability of the code.

MODULARITY:

The code is well organized into modules and classes, making it easy to understand and maintain.

USE OF TYPES

Type annotations are used to improve readability and make it easier to detect errors.

DOCUMENTATION

Each class and method has a clear description of its purpose and function.

ANTI PATTERNS

GOD OBJECT

There are no objects that do too many things. Each class has a clear responsibility.

SPAGHETTI CODE

The code is well structured and modularized, avoiding tangled and difficult-to-follow dependencies.

HARDCODING

There are no values hard-coded into the code. Data is loaded from external files.

SINGLE RESPONSIBILITY PRINCIPLE (SRP)

01

BookDAO: This class has a single responsibility, which is to represent the data for a book.

02

BookFactory: Its only responsibility is to create BookDAO instances. BookRepository: It is responsible for loading and providing book data.

03

SearchStrategy and its subclasses: Each one has the responsibility of implementing a specific search strategy.

OPEN/CLOSED PRINCIPLE (OCP):

01

SearchStrategy: The SearchStrategy base class is open for extension (you can add new search strategies) but closed for modification (you don't need to change the base class to add new strategies).

02

Catalog: You can add new search strategies without modifying the Catalog class.

LISKOV SUBSTITUTION PRINCIPLE (LSP)

01

Subclasses of SearchStrategy (SearchByAuthor, SearchByISBN, SearchByTitle) can be used in place of the base SearchStrategy class without altering the expected behavior of the program.

DEPENDENCY INVERSION PRINCIPLE (DIP):

01

Catalog depends on the SearchStrategy abstraction instead of relying on concrete implementations. This allows changing search strategies without modifying the Catalog class.

CONCLUSIONS

The Web Book Store project presents a well-structured, modular and scalable architecture, aligned with SOLID principles and supported by design patterns such as Factory Method, Strategy and Singleton, which optimize the creation, search and management of the book catalog. The application facilitates user interaction through an efficient shopping cart and an adaptable search system, while administrators can manage the product offering in an intuitive way. Its modular design based on layers (repositories, services, controllers) allows easy expansion and maintenance, ensuring that new functionalities can be incorporated without affecting the existing structure. In addition, common anti-patterns have been avoided, keeping the code clean, understandable and well documented, which guarantees its long-term maintainability.


THANK
YOU!