Софтуерна документация за проекта Библиотека

Хюлиа Мустафа, КН, група 2
ФН: 81779

Съдържание

1.Увод

1.1 Описание и идея на проекта

Настоящият документ запознава читателите с функционалностите, начините за работа, перспективите и функционалните изисквания на продукта Библиотека.

Проектът Библиотека реализира създаването на система изградена от потребители-клиенти и администратори, която позволява изпълняването на различни функционалности върху книгите, който се съдържат в една библиотека.

1.2 Цел и задачи на разработката

Настоящият документ има за цел да запознае читателите с основните функции на продукта Библиотека, като е обърнато особено внимание върху реализацията на основните методи и класове. Показани са различни алгоритми за решаване на поставените от клиента задачи.

1.3 Оформление на документа

При написването на документа е използван шрифт Times New Roman. Текстът е с размер 12.Всички заглавни редове са удебелени и с размер 18. Подзаглавията са с размер 14.

2.Преглед на предметната област

2.1 Концепции и алгоритми

Проектът Библиотека е реализиран чрез парадигмата за обектно-ориентираното програмиране, която представлява моделиране на предмети както от реалния свят, така и от научните сфери, чрез обекти, които да взаимодействат помежду си. Основни концепции, които се обхващат от ООП(обектно-ориентирано програмиране) са идеята за абстракция на данните, капсулиране(т.е. скриване на съществените данни за потребителя), полиморфизъм(обекти от един и същи тип да имат един и същи интерфейс, но различна реализация) и наследяване.

В проекта библиотека са включени всички основни концепции на ООП , чрез които се изгражда една надеждна и гъвкава система, която е по-проста за разработка и поддръжка.

В проектът е използван алгоритъмът на мехурчето за сортиране, който е с квадратична сложност.

2. 2 Функционални изисквания

2.2.1 Влизане в системата

А) Потребителят трябва да въведе потребителско име и парола

Б) Ако предоставената от потребителя информация не е валидна, системата трябва да извежда съобщение за грешка.

2.2.2 Използване на системата

А) След като потребителят е влязъл в системата, на него му се предоставя списък с всички налични функции, който той може да изпълнява.

Б) Потребителят може да намира информация за книги с персонален номер, може да търси определени книги по зададен критерий, да сортира книги по зададен критерии, да извежда последователно информация за всяка книга.

В) Ако потребителят е администратор, на него му се предоставя възможност да добавя нов потребител с име и парола и също така да премахва потребител.

Г) При разглеждане на конкретна книга се предоставя информация за заглавие, автор, жанр и персонален номер.

Основни функционалности, които трябва да бъдат предоставени на отделните класове.

А) Функционалности на класа „Потребител“.

- ✓ Има достъп до системата за вход
- ✓ Може да получава информация за всички книги
- ✓ Намира книги по зададен критерии
- ✓ Сортира книги
- ✓ Дава информация за определена книга с персонален номер

Б) Функционалности на класа „Клиент“

- ✓ Притежава същите функционалности като потребителя

В) Функционалности на класа „Администратор“

- ✓ Добавя и премахва потребител

2.3 Нефункционални изисквания

Изисквания за качеството на софтуера

2.3.1 Адаптивност

Системата трябва да бъде написана по начин, отговарящ на обшоприетите добри практики, които да позволят лесна и бърза поддръжка и промяна на интерфейса до 48 часа.

Промените в системата не трябва да водят до загуба на данни

2.3.2 Преизползваемост

Системата трябва да бъде разработена по такъв начин, че да позволява максималното преизползване на код за други системи от този тип.

3.Проектиране

3.1 Обща архитектура- ООП дизайн

Продуктът е изграден на базата на основни концепции в обектно-ориентираното програмиране, като за неговата реализация са използвани няколко основни класа „Библиотека“, „Книга“, и „Потребител“, като класът „Потребител“ се наследява от 2 класа - „Клиент“ и „Администратор“.

Всяка една книга се определя от няколко основни фактора:

- ✓ заглавие
- ✓ автор
- ✓ жанр
- ✓ описание
- ✓ година на издаване
- ✓ рейтинг
- ✓ идентификационен номер

Класът „Библиотека“ съдържа набор от книги.

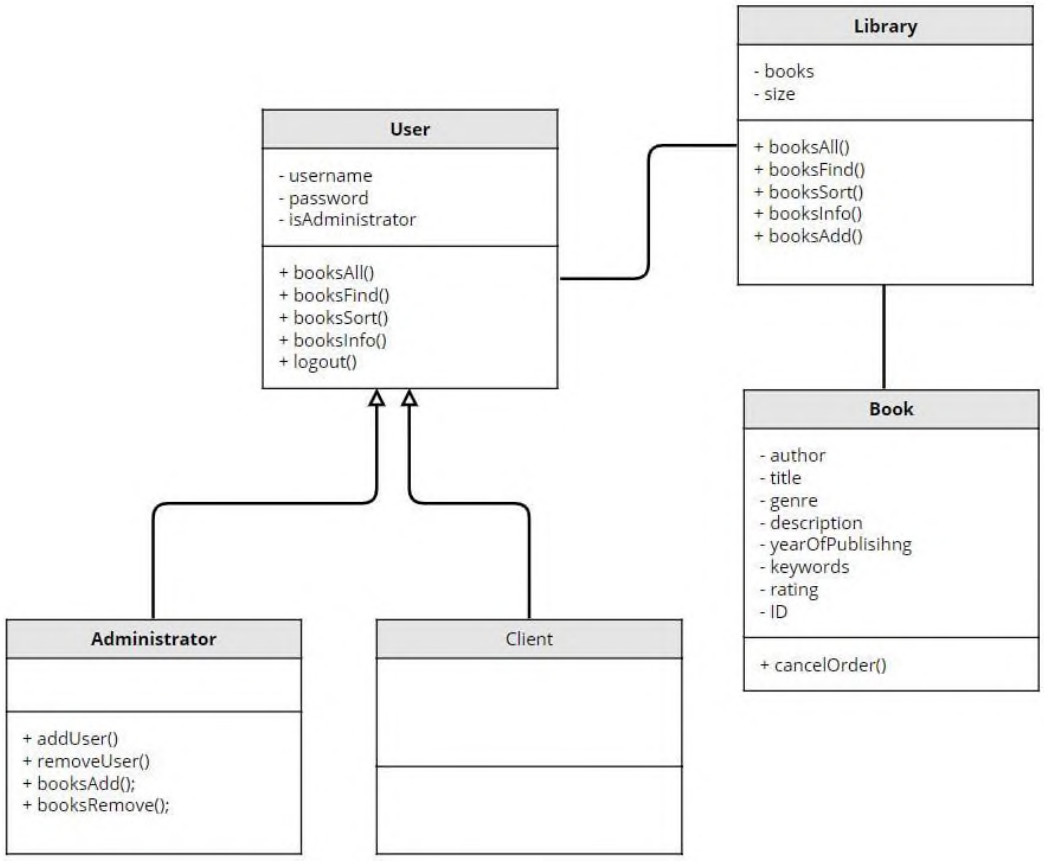
Класът „Потребител“ е изграден от:

- ✓ Потребителско име
- ✓ Парола
- ✓ Идентификатор, който указва дали потребителят е администратор или клиент

Класовете „Клиент“ и „Администратор“ са изградени на същия принцип като „Потребител“ с тази разлика, че те предоставят различни функционалности в зависимост от това дали потребителят е администратор или клиент.

3.2 UML диаграми

Приложената по-долу 1. UML диаграма показва йерархията на проекта Библиотека, като визуализира взаимоотношенията между отделните класове. Показани са основни характеристики и функционалности на всеки един отделен клас.



1. UML диаграма, показваща йерархията на проекта Библиотека

4. Реализация, тестване

4.1 Реализация на класове

Класовете са реализирани чрез обектно-ориентиран код. Класът „Книга“ е реализиран чрез, като са използвани готовите контейнери като “vector” и стринг, предоставени от стандартната библиотека [*STL(Standart Template Library)*] на езика C++.

Класът книга е реализира чрез готовите контейнери предоставени от *STL*.
Приложен е примерен код , който показва неговата деклация:

```
class Book {
private:
    std::string author;
    std::string title;
    std::string genre;
    std::string description;
    int yearOfPublishing;
    std::string keywords;
    int rating;
    int ID;

public:
```

```
    Book();
    Book(std::string authorData, std::string titleData, std::string
genreData, std::string descData, int yearData, std::string keywordsData, int
IDData);
    ~Book();


    std::string getAuthor()const{ return author;}
    std::string getTitle()const{return title;}
    std::string getGenre()const { return  genre;}
    std::string getDescription()const { return description;}
    int getYearOfPublishing()const { return yearOfPublishing;}
    std::string getKeyWords()const { return  keywords;}
    int getRating()const { return rating;}
    int getID()const { return ID;}


    void setAuthorName(std::string authorName);
    void setYearOfPublishing(int yearData);
    void setRating(int ratingData);
    void setID(int idData);



};
```

Приложен е примерен код за декларацията на класа „Потребител“ и показва някой специфични функции:

```
class User {
private:
    std::string username;
    std::string password;
    bool isAdministrator;
public:
    User();
    User(std::string nameData, std::string passwordData, bool
isAdmin);
    virtual ~User();

    std::string getUsername() const { return username;}
    std::string getPassword()const { return password;}


    virtual void booksInfo(int personalNumber, Library& lib);
```

```
    virtual void booksAll(Library& lib);
    void booksSort(std::string option, Library& lib,std::string
wayOfSorting="asc");
    virtual void booksFind(std::string option, std::string value,
Library& lib);


};
```

Класовете „Клиент“ и „Администратор“ са наследници на класа „Потребител“, като „Администратор“ добавя нова функционалност към целия проект като може да добавя и премахва както книги, така и потребители.

Класът „Библиотека“ съдържа в себе си вектор от книги, като в него са реализирани и функциите за работа с файлове.

4.2 Управление на паметта и алгоритми. Оптимизации.

Проектът е реализиран чрез контейнерите предоставени от *STL*, като е разгледа предварително реализацията на функциите, които са използвани наготово.

В реализацията на функцията, която сортира книгите по зададен критерии е използван „Метода на мехурчето“ („*Bubble Sort*“) , който има квадратична сложност. Приложен е примерен код за реализацията на „метода на мехурчето“, който сортира по възходящ ред:

```
for(size_t i=0; i<books.size()-1;i++)
{

    for(size_t j=0; j<books.size()-i-1;j++)
    {
        if(books[j].getRating()>books[j+1].getRating())
        {
            Book temp;
            temp=books[j];
            books[j]=books[j+1];
            books[j+1]=temp;
        }
    }

}
```

Алгоритъмът, може да се оптимизира чрез използването на друг алгоритъм за сортиране с по-малка сложност.

5.Заключение

5.1 Обобщение на изпълнението на началните цели

Проектът реализира по-голямата функционалност от изискванията на клиента.

5.2 Бъдещи перспективи на продукта

Тук се предоставят подобрения, свързани с бъдещата перспектива на продукта:

Проектът Библиотека ще предоставя възможност за влизане в системата с биометрични данни, с цел потвърждаване на самоличността на клиента. Ще бъдат разработени 3 начина за идентифициране:

а) Както потребителят, така и администраторът ще могат да въвеждат своят пръстов отпечатък в системата, при по-нататъчни влизания няма да се налага да използват паролата си.

б) Ще бъде въведено гласово разпознаване, чрез което всеки потребител ще може да влезе в системата с определена от него ключова дума.

в) Ще се предоставя възможност за сканиране на ириса, с цел улесняване на влизането в системата.