НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Кафедра інформаційних систем та технологій

Курсова робота

З дисципліни «Програмування – 1. Основи програмування»
Тема: Бібліотека

Керівник
ст. викл. Яленецький В.А
«Допущений до захисту»
_____
(Особистий підпис керівника)
« » _____2022р.
Захищений з оцінкою
_____
(оцінка)

Виконавець
ст. Сергєєв В.В.
залікова книжка № ІС–1126
гр. ІС-11
_____
(особистий підпис виконавця)
« »_____2022 р.

Члени комісії:

_____ _____
(особистий підпис) (розшифровка підпису)

_____ _____
(особистий підпис) (розшифровка підпису)

Київ – 2022

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
(назва навчального закладу)
Кафедра ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Дисципліна «Програмування – 1. Основи програмування»

Курс 1 Група ІС-11 Семестр 2

ЗАВДАННЯ

на курсову роботу студента

Сергєєва Владислава Віталійовича

1. Тема роботи: Бібліотека

2. Строк здачі студентом закінченої роботи 22.06.2022

3. Вихідні дані до роботи: 06.06.2022

Мова програмування – Інструментарій: мова C#; середовище Visual Studio; інтерфейс WinForm, фреймворки; Цільові вимоги: зберігати списки читачів, книг та авторів у JSON файлах; Кожен читач може взяти та повернути книгу; Генерування списку боржників; Додавання та видалення читача зі списку користувачів бібліотеки; Пошук в переліку читачів по ключовому слову

4. Зміст розрахунково–пояснювальної записки (перелік питань, що підлягають розробці)

1) Розглянути існуючі рішення та визначити сутності об'єкта розробки.
2) Розробити класи однотипних об’єктів (4 класи);
3) Визначити кожній сутності її атрибути та ключ;
4) Побудувати діаграму зв’язків «сутність – зв’язок» (ER–діаграма);
5) Запрограмувати функціонал та інтерфейс мінімально-працездатного застосунку
6) Сформувати інструкцію користувача мінімально-працездатного застосунку
7) Протестувати мінімально-працездатний застосунок

Додатки: Вихідний код програми; Приклад виконання застосунку

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Діаграми прецедентів, архітектури системи, скріншоти програми;

6. Дата видачі завдання: 13.04.2022

КАЛЕНДАРНИЙ ПЛАН

№, п/п	Назва етапів виконання курсової роботи	Строк виконання	Підписи або

		етапів роботи	**примітки**
1.	Отримання теми курсової роботи	13.04.22	
2.	Формування вимог	15.04.22	
3.	Проектування	17.04.22	
4.	Реалізація	29.04.22	
5.	Тестування	10.05.22	
6.	Підготовка пояснювальної записки	14.06.22	
7.	Здача курсової роботи на перевірку	22.06.22	
8.	Захист курсової роботи	22.06.22	

Студент_____ Сергєєв ВЛАДИСЛАВ

(підпис)

Керівник_____ Валерій ЯЛЕНЕЦЬКИЙ

(підпис)

«____»_____ 2022 р

АНОТАЦІЯ

Сергєєв Владислав Бібліотека. КПІ ім. Ігоря Сікорського, Київ, 2022.

Робота містить 38с. тексту, 21 рисунок, 9 посилань на додатки.

Ключові слова: бібліотека, користувач, оцінка, книга, вчитель, C#, клас, об’єктно-оріентоване програмування.

Об’єктом розробки є застосунок для адміністрування бібліотеки.

Мета розробки – полегшення адміністрування процесів бібліотеки.

У курсовій роботі розроблено застосунок для адміністрування процесів бібліотеки. Проведено ретельний аналіз та вибір системи і визначено її особливості. Систему розроблено на мові програмування C#. Взято до уваги особливості написання застосунку згідно з парадигмами об’єктно-орієнтованого програмування.

Отримані результати можуть бути корисними при розробці аналогічних чи подібних застосунків. Отримані навички можна подалі застосовувати в області веб-програмування

ЗМІСТ

ВСТУП 6

РОЗДІЛ 1. Проектування ПЗ 7

1.1. Сценарій роботи програми 7

1.2. Документація 8

1.2. ER-діаграма 13

1.3. Діаграма класів 14

РОЗДІЛ 2. Програмне забезпечення 15

1.1 Структура проекту 15

1.2 Програмний код 16

РОЗДІЛ 3. Інструкція користувачу 20

ВИСНОВКИ 25

ДЖЕРЕЛА 26

ДОДАТОК 27

ВСТУП

В 21 столітті технології розвиваються дуже швидко, все навколо нас починає автоматизуватися, складні системи потребують застосунків для адміністрування процесів в них.

Актуальність теми полягає у адмініструванні процесів у державних та приватних установах. Замість великої купи записів на папері адміністратор застосунку може вносити дані до електронної системи. Це значно спрощує та пришвидшує роботу працівників.

За предмет дослідження взято бібліотеку. На її прикладі можна наглядно дослідити процеси та вирішити основні задачі, які стоять перед розробником, а саме: редагування каталогу книг, користувачів, авторів; видача та повернення книги; пошук серед каталогу по ключовим словам; створення користувацької сторінки, де користувач зможе взяти, повернути книги та переглянути каталог книг, які він читає та каталог усіх доступних книг.

Результатом виконання курсової роботи має стати готовий застосунок для адміністрування процесів в бібліотеці.

Курсова робота складається з наступних розділів: вступ, основні розділи, висновки, список використаних джерел із 9 найменувань, 1 додатку. графічна частина включає 21 фотографій. Загальний обсяг ..сторінок

РОЗДІЛ 1. Проектування ПЗ

1.1. Сценарій роботи програми

Користувач має смогу авторизуватися(зареєструватися), щоб отримати доступ до коталогу бібліотеки. Після входу в систему користувач переходить на сторінку, де він може переглядати всі доступні книги бібліотеки. При виборі конкретної книги користувач може взяти її на користування, після цього обрана книга з’явиться в списку книг користувача. Коли читач успішно скористався, він має змогу повернути книгу. Також користувачу доступна функція пошуки книги по ключовим словам в назві книги.

Користувач з правами адміністратора після авторизації переходить на сторінку де отримує можливість додавати, редагувати та видаляти книги, читачів та авторів. При виборі конкретної книги відкривається поле, з інформацією про читача в користуванні якого перебуває книга. Також, при виборі читача зі списку всіх наявних читачів з’являється інформація про книги, якими він користується.

Після вдалої сесії користувач має змогу вийти з облікового запису.

Сценарій роботи програми зображений на Рисунку 1.1:

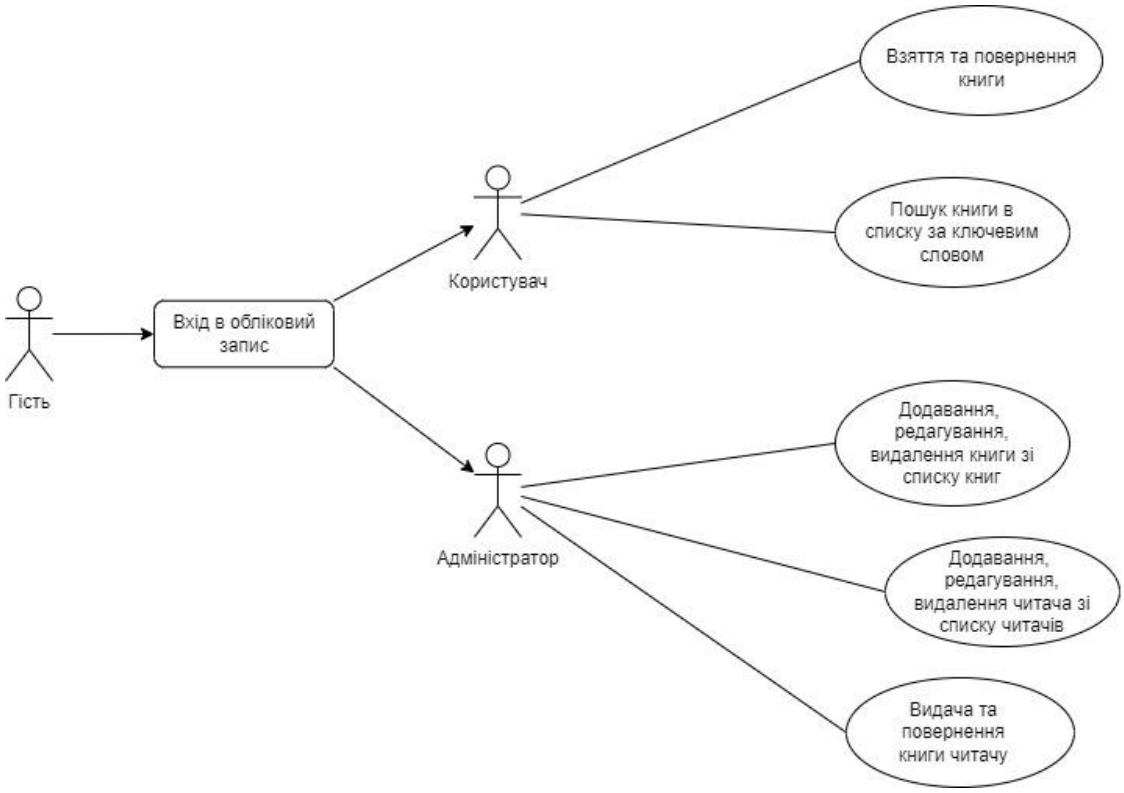

Рисунок 1.1.1

1.2. Документація

Список класів:

1) Library – головний клас програми, який в собі містить список книг, користувачів та авторів:
    a) Поля:
        - BooksList{}
        - UsersList{}
        - AuthorsList{}
    b) Конструктори
        - Library() – конструктор, який з бази даних заповнює всі поля класу .
    c) Методи

- BaseFirstInitalisation() – метод, який заповнює даними з бази даних поля: BooksList{}, UserList{}, AuthorsList{}.
- SignIn() – метод, що виконує авторизацію користувача
- GetUserById() – метод, що приймає id-користувача, а повертає користувача з цим Id

2) Person – абстрактний клас, що надає інформацію про ім`я користувача та метод для сортування студентів по групам
    a) Поля
        - Name
        - DateOfBirth
3) User – клас-нащадок від Person, що надає інформацію про зареєстрованого користувача, та методи для роботи з ним

    a) Поля
        - Name
        - Password
        - IsAdmin – булеве значення
        - _Id – унікальний ідентифікаційний номер
        - BooksInReadingIds[] – перелік id-номерів книг, які користувач читає
        - IsReading - булева змінна, що означає перебування книги в користуванні чи ні
        -
    b) Конструктори
        - User(name, password, isAdmin) – створює екземпляр класу User з вказаними даними
    c) Методи

- ToString() – метод, що повертає строку, представляючу цей клас

4) Author – клас-нащадок від Person, що надає інформацію про автора книг
    a) Поля
        - Name
        - DateOfBirth
        - _Id – унікальний ідентифікаційний номер
        - BooksIds[] - перелік id-номерів книг, які автор написав
    b) Конструктори
    c) Author(username,password,name,group_id) – створює об’єкт класу Author з вказаними даними

    d) Методи
        - ToString() – метод, що повертає строку, представляючу цей клас
5) Book – клас, що надає інформацію про книгу
    a) Поля
        - Title
        - Author_Id – id-номер автора книги
        - _Id – унікальний id-номер
        - IsOnUse – булеве значення, що позначає перебування книги у користуванні читача
        - Reader_Id – id-номер читача книги
    b) Конструктори
        - Book(title, author_id) – додає новий об’єкт до класу Book
    c) Методи
        - ToString() – метод, що повертає строку, представляючу цей об’єкт

Список форм:

1) AuthFrom – форма, що відповідає за реєстрацію та авторизацію користувачів
    a) Методи
        - SignInBtn_Click() – передає дані з текстових полів на серверну частину. В разі успішного виконання надає доступ до користувацької або адміністраторської форми. В разі невдачі – пропонує спробувати вхід ще раз або зареєструватися
        - RegBtn_Click() - передає дані з текстових полів на серверну частину. В разі успішного виконання надає доступ до користувацької або адміністраторської форми. В разі невдачі – пропонує спробувати вхід ще раз
2) AdminForm – форма, що відповідає за додавання, редагування та видалення нових книг, користувачів та авторів
    a) Методи
        - UserForm () – метод, що викликається при першому запуску форми, який в свою чергу ініціалізує компоненти форми
        - FormLists_Load() – метод, який викликається при завантаженні FormLists(полів для виведення списку книг, користувачів та авторів).Метож задає значення цих полів.
        - RefreshBooksList() та RefreshReadersList() – методи, які оновлюють зміст поля зі списком книг та читачів відповідно.
        - AddBookBtn_Click() AddReaderBtn_Click() – методи, які беруть дані з текстових комірок та передають їх у методи створення книги та читача відповідно, Викликаються при натисканні на відповідні кнопки
        - EditBookBtn_Click() та EditUserBtn_Click() - методи, які беруть дані з текстових комірок та передають Їх у методи редагування книги та читача відповідно, Викликаються при натисканні на відповідні кнопки

- RemoveBookBtn_Click() та RemoveBookBtn_Click() - методи, які беруть дані з текстових комірок та передають Їх у методи видалення книги та читача відповідно, Викликаються при натисканні на відповідні кнопки
- BooksListBox_SelectedIndexChanged() та ReadersListBox_SelectedIndexChanged – методи, функція яких, підставити дані про книги чи користувача у відповідні текстові комірки. Викликається при натисканні мишею на конкретне поле в ListBox
- ShowUserBooks(reader) – метод приймає об’єкт читача. Формує та демонструє список книг відповідно до іх id-номерів в полі конкретного користувача(reader)
- ShowBookReaders(book) – метод приймає об’єкт книги. Демонструє користувача, який читає книгу
- LogOutBtn_Click() – метод закриває користувацьку форму та відкриває форму авторизації. Викликається при натисканні на відповідну кнопку.

3) UserForm - форма, що відповідає за обслуговування користувачів (видача та повернення книг)
   a) Методи
      - UserForm() – метод приймає об’єкт користувача. Відповідає за заповнення полів форми відповідно до даних користувача.
      - BorrowBook_Click() – метод, який надсилає запит на серверну частину про надання користувачу книги.
      - ReturnBook_Click() - метод, який надсилає запит на серверну частину про повернення книги від користувача.
      - SerchBtn_Click() – метод, який при натисканні на кнопку пошуку виконує пошук по ключовому слову, та запускає процес

оновлення ListBox(списку книг користувача) відповідно до виконаного пошуку

1.2. ER-діаграма

ER-діаграму зображено на Рисунку 1.2.1

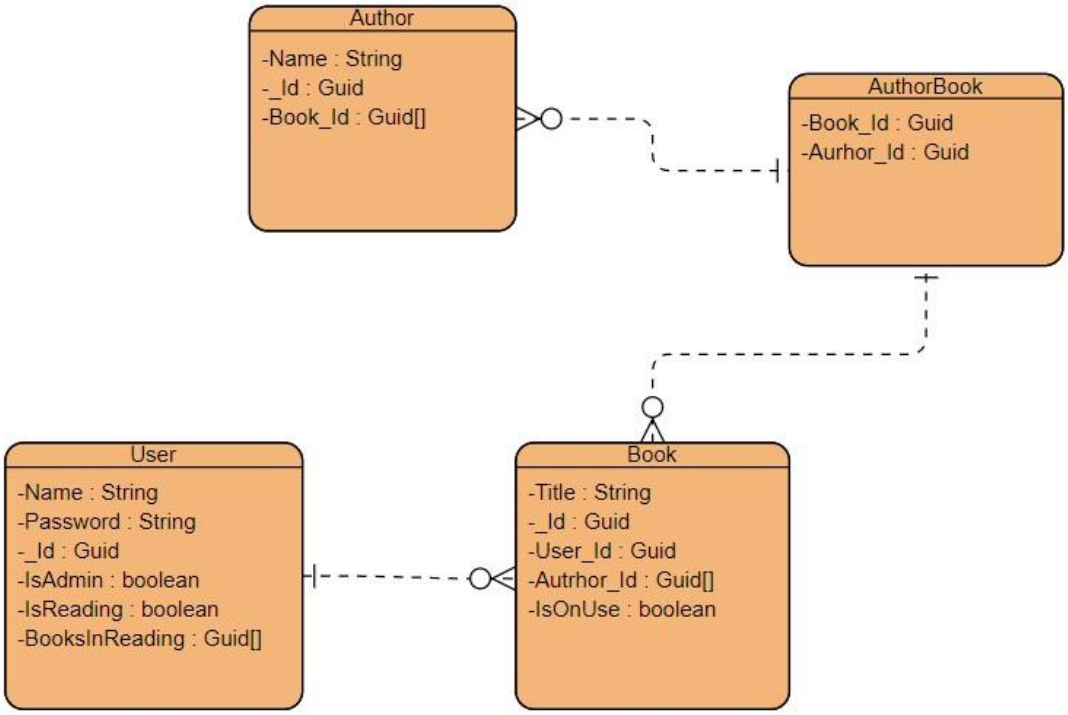

Рисунок 1.2.1 – ER-діаграма

На діаграмі зображено схему зв’язків між класами. В програмі використано три види зв’язку: один до одного (1:1), один до багатьох (1:*) та багато до багатьох (*:*).

1) Один до одного (1:1) – такий тип зв’язку побудовано між класом Book та класом User. З логіки програми випливає, що у однієї книги (Book) може бути лише один користувач (User)
2) Один до багатьох (1:*) – такий тип зв’язку побудовано між класом User та класом Book. Удного користувача може бути більше однієї книги.
3) Багато до багатьох (*:*) – такий тип зв’язку побудований між класами Book та Author через клас розв’язку AuthorBook. З логіки програми

видно, що книга може мати більше одного автора та автор може написати більше однієї книги.

1.3. Діаграма класів

На рисунку 1.3.1 зображено діагруму класів програми

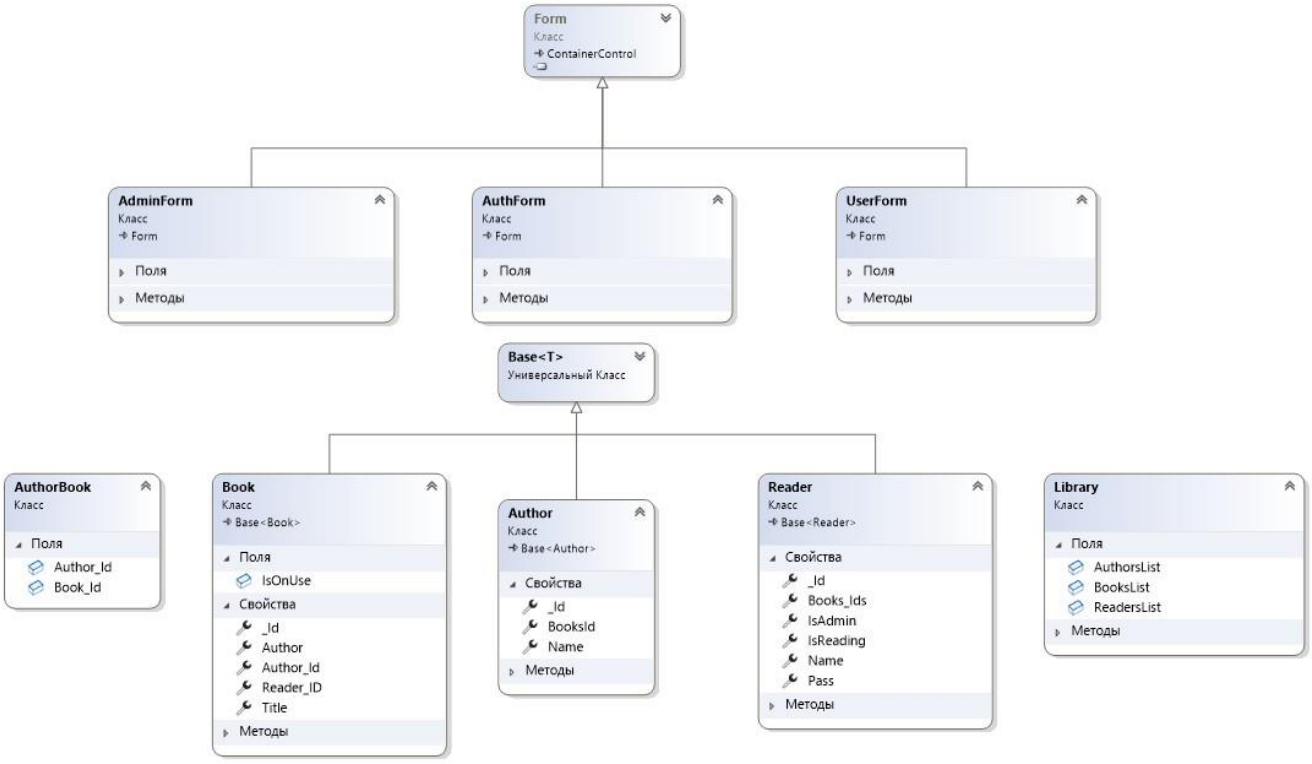

Рисунок 1.3.1- Діаграма класів

На діаграмі зображені класи програми, зображено властивості, методи та поля класів. Також позначено наслідування

РОЗДІЛ 2. Програмне забезпечення

1.1 Структура проекту

На Рисунку 1.1.1 зображено структуру проекту

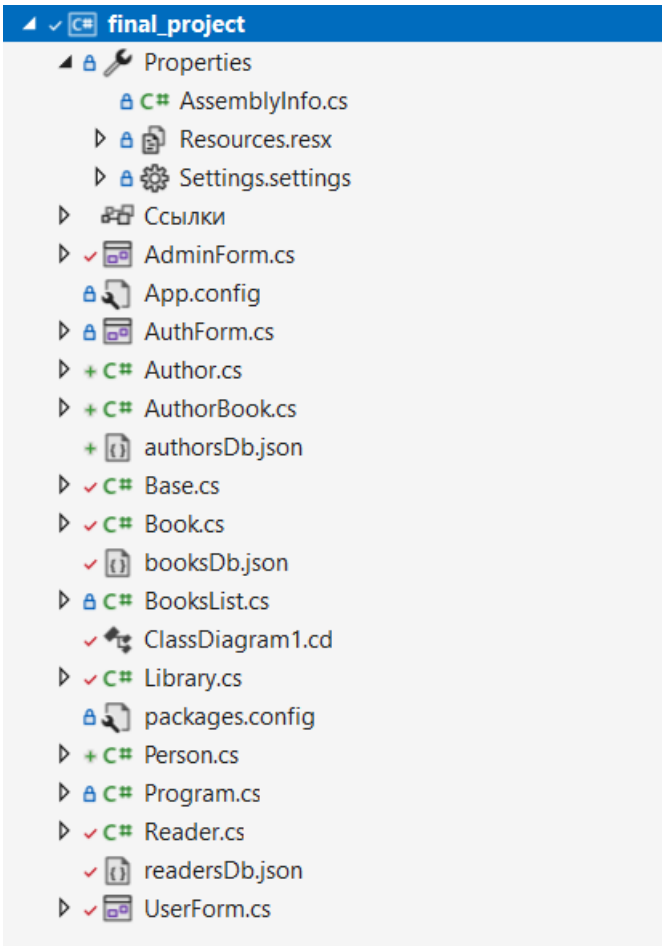

Рисунок 1.1.1

1.2 Програмний код

На Рисунку 1.2.1 зображено клас Reader з його властивостями та методами

```
internal class Reader : Base<Reader>
{
    Ссылок: 10
    public string Name { get; set; }
    Ссылок: 0
    public bool IsReading { get; set; }

    Ссылок: 7
    public Guid _Id { get; set; }
    Ссылок: 5
    public string Pass { get; set; }
    Ссылок: 5
    public bool IsAdmin { get; set; }
    Ссылок: 6
    public List<Guid> Books_Ids { get; set; }

    Ссылок: 0
    public Reader()...
    ссылка: 1
    public Reader(string name, string pass, bool isAdmin)
    {
        Console.WriteLine("bookCreated");
        Name = name;
        Pass = pass;
        IsAdmin = isAdmin;
        Books_Ids = new List<Guid>();
        Books_Ids.Add(Guid.NewGuid());
        _Id = Guid.NewGuid();
    }
    Ссылок: 0
    public override string ToString()
    {
        return $"{Name} {Pass} {(IsAdmin ? "Admin" : "")}";
    }

    Ссылок: 0
    ~Reader()
    {
        Console.WriteLine("Reader deleted!");
    }
}
```

Рисунок 1.2.1

На Рисунку 1.2.2 зображено клас Book з його властивостями та методами

```
Ссылок: 14
internal class Author
{
    Ссылок: 4
    public string Name { get; set; }
    Ссылок: 5
    public Guid _Id { get; set; }
    Ссылок: 0
    public List<Guid> BooksId { get; set; }

    Ссылок: 0
    public Author()...
    ссылка: 1
    public Author(string name)
    {
        Console.WriteLine("AuthorCreated");
        Name = name;
        _Id = Guid.NewGuid();
    }
    Ссылок: 0
    public override string ToString()
    {
        return $"{Name}";
    }
}
```

Рисунок 1.2.2

На рисунку 1.2.3 зображено клас Book з його властивостями та методами

```
internal class Book
{
    Ссылок: 5
    public string Title { get; set; }

    Ссылок: 4
    public string Author { get; set; }

    Ссылок: 7
    public Guid _Id { get; set; }

    Ссылок: 3
    public Guid Reader_ID {get; set;}

    public bool IsOnUse = false;
    ссылка: 1
    public Guid Author_Id { get; set; }
    Ссылок: 0
    public Book()...
    ссылка: 1
    public Book(string title, string author, Guid authorId)
    {
        Console.WriteLine("BookCreated");
        Title = title;
        Author = author;
        Author_Id = authorId;
        _Id = Guid.NewGuid();
    }

    Ссылок: 0
    public override string ToString()
    {
        return $"{Title}  {Author}" ;
    }
    Ссылок: 0
    ~Book()
    {
        Console.WriteLine("Book deleted");
    }
}
```

Рисунок 1.2.3

Методи створення нової книги, кориистувача таавтора зображенно на Рисунку 1.2.5

```
public void AddNewBook(string title, string author, Guid authorId)
{
    Book book = new Book(title, author, authorId);
    BooksList.Add(book);
    RefreshBooksJson();
}

Ссылок: 2
public Guid AddNewUser(string name, string pass, bool isAdmin)
{
    Reader reader = new Reader(name, pass, isAdmin);
    ReadersList.Add(reader);
    RefreshReadersJson();
    return reader._Id;
}
Ссылок: 2
public Guid AddNewAuthor(string name)
{
    Author author = new Author(name);
    AuthorsList.Add(author);
    RefreshAuthorsJson();
    return author._Id;
}
```

Рисунок 1.2.5

Метод SignIN() продемонтсровано на Рисунку 1.2.6

```
public Reader SignIn(string username, string password)
{
    Console.WriteLine(ReadersList[0].Name);
    Reader reader = ReadersList.Find(x => x.Name == username && x.Pass == password);
    if (reader == null)
    {
        Console.WriteLine("SignIn failed");
        return null;
    }
    else
    {
        Console.WriteLine(reader.Name);
        return reader;
    }
}
```

Рисунок 1.2.6

Методи пошуку книги та користувача за їхніми id-номерами зображено на Рисунку 1.2.7

```
public Reader GetUserById(Guid id)
{
    return ReadersList.Find(x => x._Id == id);
}

ссылка: 1
public Book GetBookById(Guid id)
{
    return BooksList.Find(x => x._Id == id);
}
```

Рисунок 1.2.7

Методи GiveBookToUser() та ReturnBookFromUser() проденстровано на Рисунку 1.4.8

```
public void GiveBookToUser(Book book, Reader reader)
{
    Console.WriteLine("give book");
    reader.Books_Ids.Add(book._Id);
    book.Reader_ID = reader._Id;
    book.IsOnUse = true;
    RefreshReadersJson();
    RefreshBooksJson();
}
ссылка: 1
public void ReturnBookFromUser(Book book, Reader reader)
{
    reader.Books_Ids.Remove(book._Id);
    book.Reader_ID = Guid.Empty;
    book.IsOnUse = false;
    RefreshReadersJson();
    RefreshBooksJson();
}
```

Рисунок 1.2.8

На Рисунку 1.2.9 зображено методи для обробки натискання на конкретний елемент списку кииг, користувачів, авторів

```
// Обробники події натискання на конкретний елемент в списку
ссылка: 1
private void ReadersListBox_SelectedIndexChanged(object sender, EventArgs e)
{
    if (ReadersListBox.SelectedItem != null)
    {
        Reader reader = ((Reader)ReadersListBox.SelectedItem);
        UserNameInput.Text = reader.Name;
        UserPassInput.Text = reader.Pass;
        IsAdminCheckBox.Checked = reader.IsAdmin;

        ShowUserBooks(reader);
    }
}

ссылка: 1
private void BooksListBox_SelectedIndexChanged(object sender, EventArgs e)
{
    if (BooksListBox.SelectedItem != null)
    {
        Book book = ((Book)BooksListBox.SelectedItem);
        BookTitleInput.Text = book.Title;
        BookAuthorInput.Text = book.Author;

        ShowBookReader(book);
    }
}

ссылка: 1
private void AuthorsListBox_SelectedIndexChanged(object sender, EventArgs e)
{
    if (AuthorsListBox.SelectedItem != null)
    {
        Author author = ((Author)AuthorsListBox.SelectedItem);
        AuthorNameInput.Text = author.Name;

        //BookAuthorNameInput.Text = author.Name;
    }
}
```

Рисунок 1.4.2

РОЗДІЛ 3. Інструкція користувачу

При запуску програми бачимо форму реэстрації/авторизації. Якщо облікового запису не існує, є можливість його створити. При вході в систему користувача з правами адміністратора переходимо на сторінку адміністратора. Якщо користувач не має таких прав він потрапляє на користувацьку сторінку

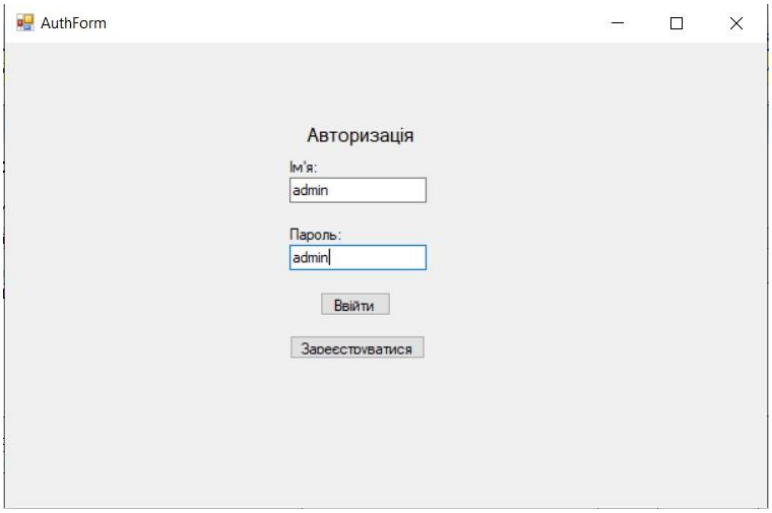

Рисунок 3.1

Після входу користувач потравляє на сторінку адміністратора

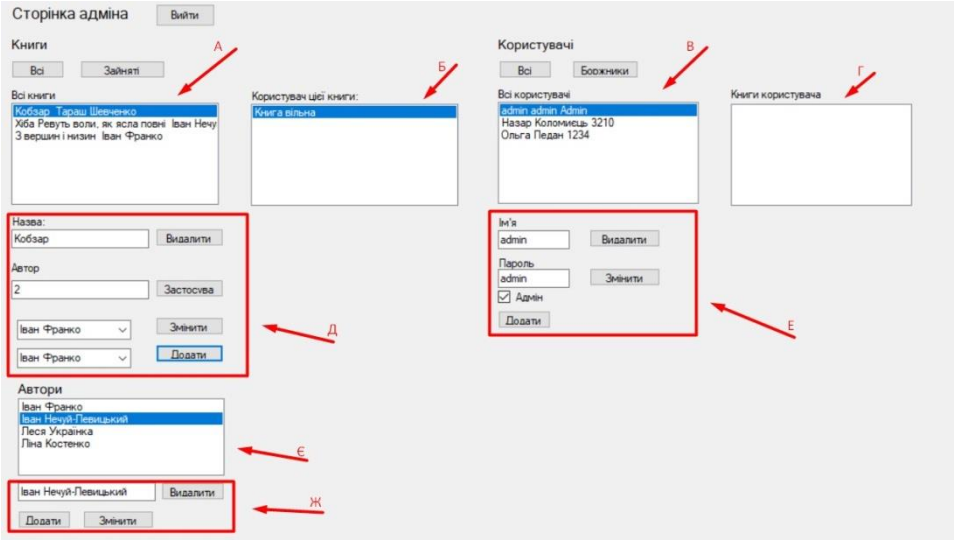

Рисунок 3.2(А-перелік всіх книг; Б-користувач обраної книги; В-перелік всіх користувачів; Г-перелік книг конкретного користувача;Д,Е,Ж-функціонад для редагування переліку книг, користувачів та авторів)

Щоб додати нову книгу користувач має написати назву книги та обрати автора з переліку, перед цим визначити кількість авторів книги, та заповнити іх.

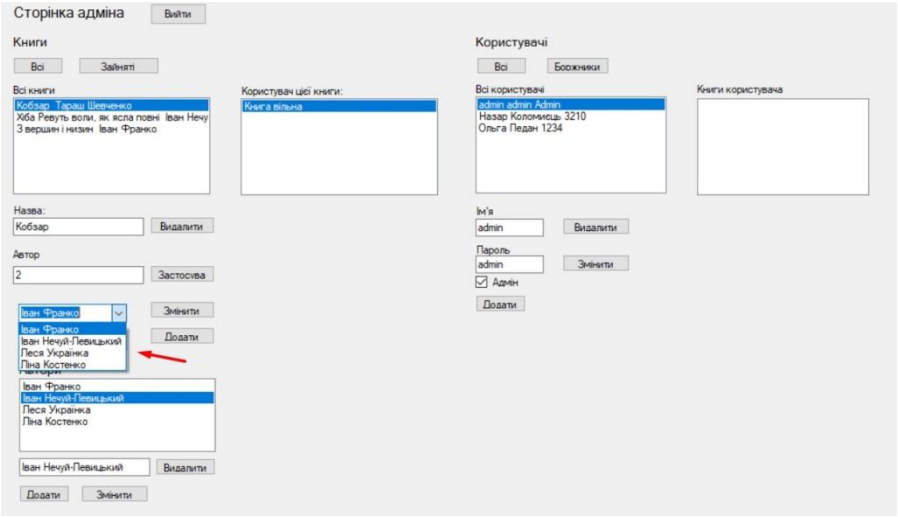

Рисунок 3.3

Щоб створити нового користувача потрібно ввести відповідні дані до текстових полів та натиснути кнопку «Додати». Аналогічно для додавання книг та авторів.

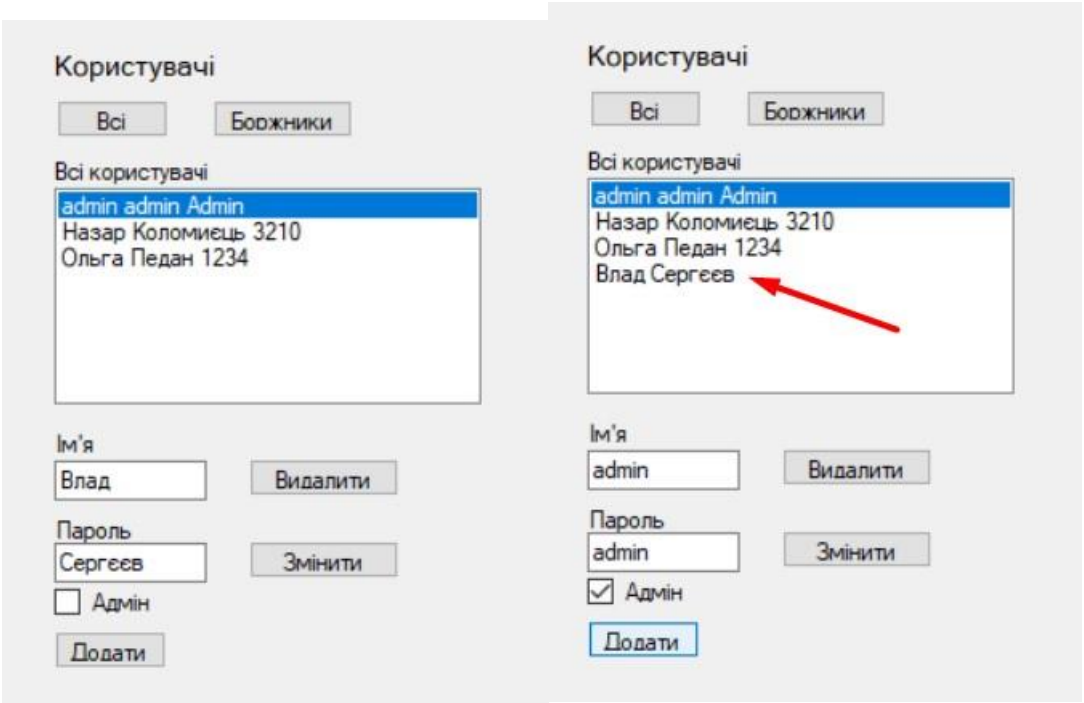

Рисунок 3.4

Якщо користувач візьме книгу в полі «Користувач цієї книги» з’явиться інформація про нього. Якщо ж ніхто не користується книгою в полі буде надпис «Книга вільна»

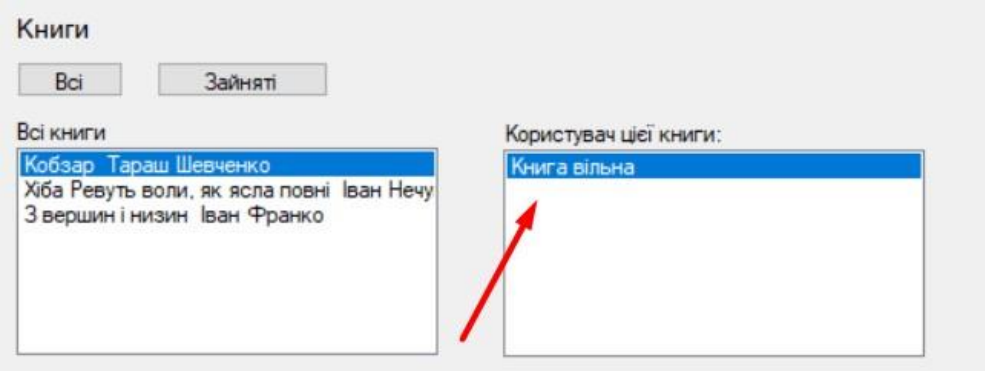

Рисунок 3.5

Аналогічно в полі «Книги користувача» з’явиться інформація про книги якими користується користувач.

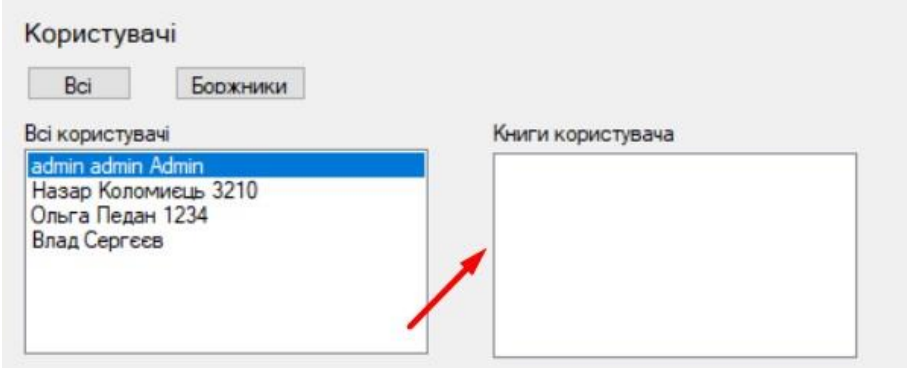

Рисунок 3.6

Для виходу з облікого запису потрібно натиснути кнопку «Вийти»

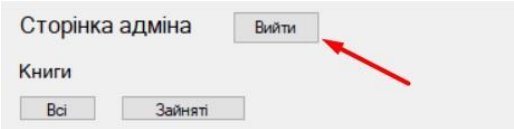

Рисунок 3.7

При вході як звичайний з правами звичайного користувача користувач потрапляю на сторінку, де видно перелік всіх доступних книг.

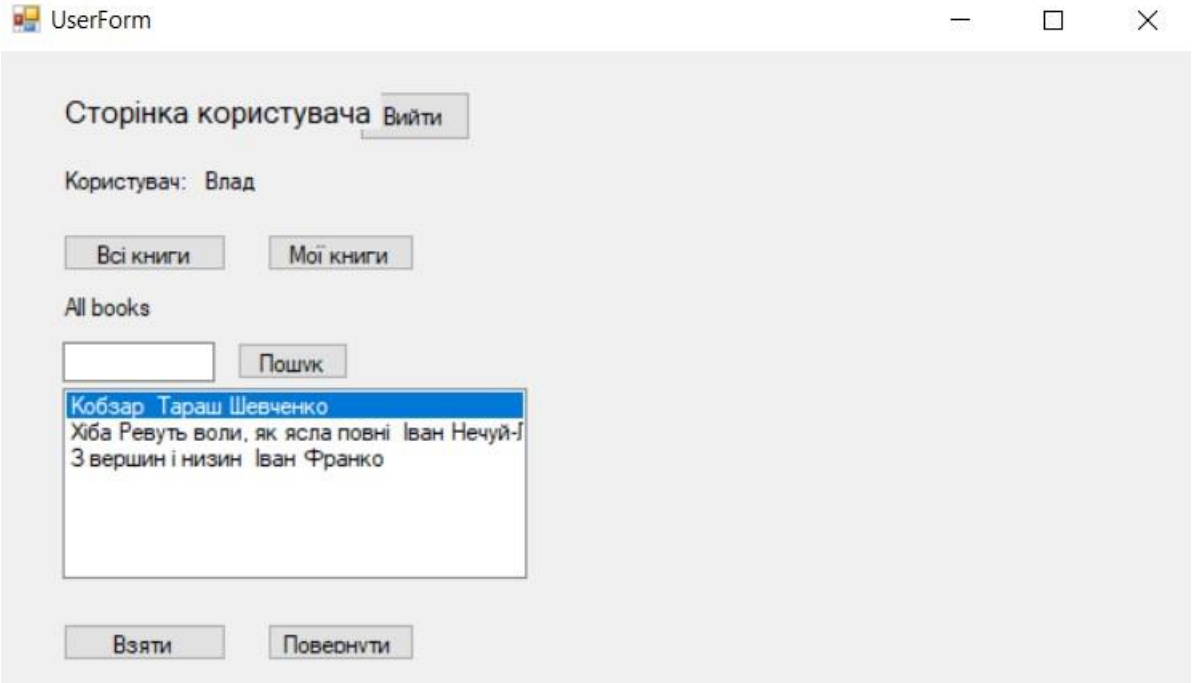

Рисунок 3.8

Щоб взяти книгу в користування треба обрати її з переліку та натиснути кнопку «Взяти».

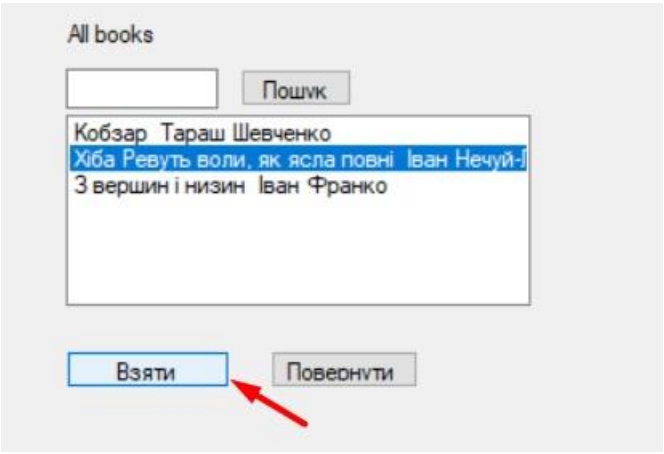

Рисунок 3.9

Щоб виконати пошук по ключовому слову треба його ввести у відповідне поле та натиснути конику «Пошук»

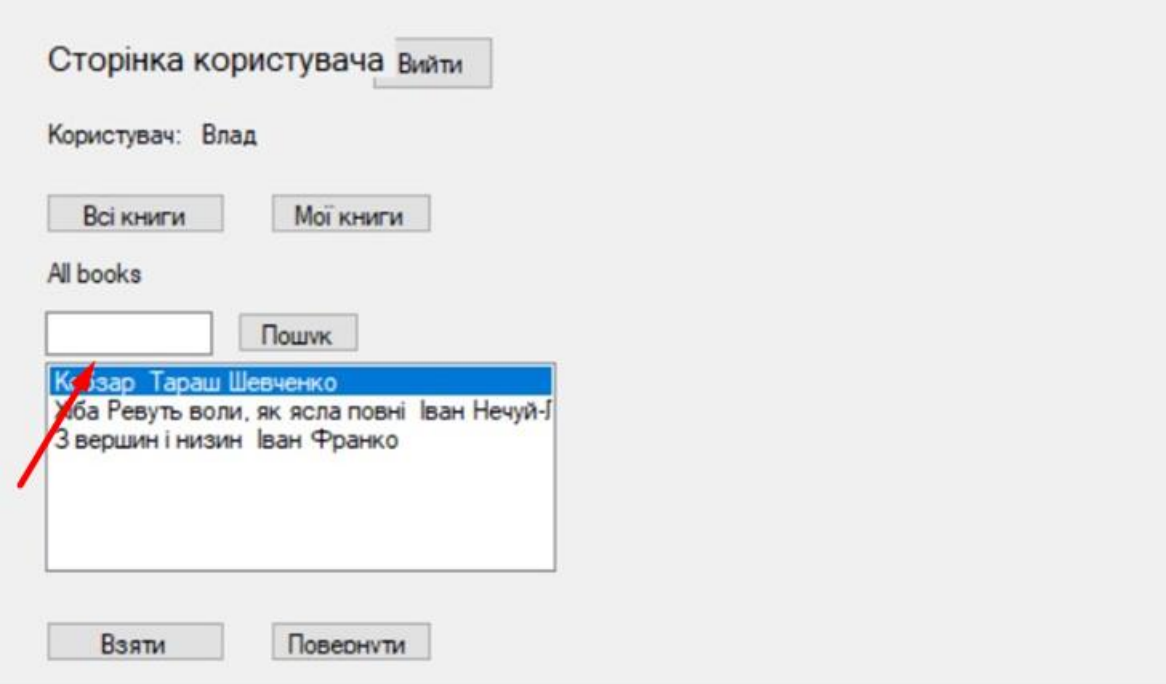
Сторінка користувача
Вийти
Користувач: Влад
Всі книги
Мої книги
All books
Пошук
Взяти
Повернути

Рисунок 3.1

ВИСНОВКИ

Результатом виконання курсової роботи став застосунок для адміністрування процесів у бібліотеці. Впровадження даного застосунку значно спрощує роботу бібліотеки.

На практиці застосовано основні методи об’єктно-оріентованого програмування для вирішення промислових задач. Показано принципи адміністрування процесів.

Першим кроком були сформульовані функціональні та нефункціональні фимого до системи, обрана мова програмування та середа розробки.

Другом кроком була розробка сценаріїв використання, проектування класів програми, побудова загальної архітектури застосунка та проектування зв’язків між сутностями

Фінальним кроком стала розробка програмного коду

ДЖЕРЕЛА

1. Introdaction to classes [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/types/classes

2. UML Use Case Diagrams. [Електронний ресурс] – Режим доступу до ресурсу: https://www.uml-diagrams.org/use-case-diagrams.html

3. Draw.io [Електронний ресурс] – Режим доступу до ресурсу: https://app.diagrams.net/

4. Online ERD Tool [Електронний ресурс] – Режим доступу до ресурсу: https://online.visual-paradigm.com/diagrams/features/erd-tool/

5. Database relationships [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/docs/en/control-desk/7.6.0?topic=structure-database-relationships

6. Github репозиторій з кодом застосунка [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/vladserheev/final_project

7. Class Diagram [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Class_diagram

8. Introdaction JSON [Електронний ресурс] – Режим доступу до ресурсу: https://www.json.org/json-en.html

9. Windows Forms app [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/visualstudio/ide/create-csharp-winform-visual-studio?view=vs-2022

ДОДАТОК

1) AdminForm.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace final_project
{
   public partial class AdminForm : Form
   {
      Library lib = new Library();
      public AdminForm()
      {
         InitializeComponent();
         RefreshBooksListBox();
         RefreshReadersListBox();
         RefreshAuthorsListBox();
      }
      public void FormLists_Load(object sender, EventArgs e)
      {
        BooksListBox.DataSource = lib.BooksList.ToList();
        ReadersListBox.DataSource = lib.ReadersList.ToList();
        AuthorsListBox.DataSource = lib.AuthorsList.ToList();
      }

      //Оновлення списків з книгами, користувачами та аторами

      public void RefreshBooksListBox()
      {
         BooksListBox.DataSource = lib.BooksList.ToList();
      }
      public void RefreshReadersListBox()
      {
         ReadersListBox.DataSource = lib.ReadersList.ToList();
      }
```

```
public void RefreshAuthorsListBox()
{
   AuthorsListBox.DataSource = lib.AuthorsList.ToList();
}

//Додавання, редагування та видалення книг, користувачів та авторів
public void AddBookBtn_Click(object sender, EventArgs e)
{
   if (AuthorsListBox.SelectedItem != null)
   {
      Author author = ((Author)AuthorsListBox.SelectedItem);
      string title = BookTitleInput.Text;
      string author1 = BookAuthorInput.Text;

      lib.AddNewBook(title, author1, author._Id);
      RefreshBooksListBox();
   }
}
public void AddReaderBtn_Click(object sender, EventArgs e)
{
   string name = UserNameInput.Text;
   string pass = UserPassInput.Text;
   bool isAdmin = IsAdminCheckBox.Checked;
   lib.AddNewUser(name, pass, isAdmin);
   RefreshReadersListBox();
}

public void AddAuthorBtn_Click(object sender, EventArgs e)
{
   Console.WriteLine("ADD BOOK");
   string name = AuthorNameInput.Text;
   lib.AddNewAuthor(name);
   RefreshAuthorsListBox();
}
```

```
public void EditUserBtn_Click(object sender, EventArgs e)
{
   if (ReadersListBox.SelectedItem != null)
   {
      Reader reader = ((Reader)ReadersListBox.SelectedItem);
      reader.Name = UserNameInput.Text;
      reader.Pass = UserPassInput.Text;
      reader.IsAdmin = IsAdminCheckBox.Checked;
      lib.RefreshReadersJson();
      RefreshReadersListBox();
   }
}

public void EditBookBtn_Click(object sender, EventArgs e)
{
   if (BooksListBox.SelectedItem != null)
   {
      Book book = ((Book)BooksListBox.SelectedItem);
      book.Title = BookTitleInput.Text;
      book.Author = BookAuthorInput.Text;
      lib.RefreshBooksJson();
      RefreshBooksListBox();
   }
}

public void EditAuthorBtn_Click(object sender, EventArgs e)
{
   if (AuthorsListBox.SelectedItem != null)
   {
      Author author = ((Author)AuthorsListBox.SelectedItem);
      author.Name = AuthorNameInput.Text;
      //lib.RefreshReadersJson();
      RefreshAuthorsListBox();
   }
}
```

```
public void RemoveBookBtn_Click(object sender, EventArgs e)
{
   if (BooksListBox.SelectedItem != null)
   {
      Book book = ((Book)BooksListBox.SelectedItem);
      Guid id = book._Id;
      lib.RemoveBook(id);
      RefreshBooksListBox();
   }
}

public void RemoveReaderBtn_Click(object sender, EventArgs e)
{
   if (ReadersListBox.SelectedItem != null)
   {
      Reader reader = ((Reader)ReadersListBox.SelectedItem);
      Guid id = reader._Id;
      lib.RemoveReader(id);
      RefreshReadersListBox();
   }
}

public void RemoveAuthorBtn_Click(object sender, EventArgs e)
{
   if (AuthorsListBox.SelectedItem != null)
   {
      Author author = ((Author)AuthorsListBox.SelectedItem);
      Guid id = author._Id;
      lib.RemoveAuthor(id);
      RefreshAuthorsListBox();
   }
}
// Обробники події натискання на конкретний елемент в списку
public void ReadersListBox_SelectedIndexChanged(object sender, EventArgs e)
{
   if (ReadersListBox.SelectedItem != null)
```

```
    {
        Reader reader = ((Reader)ReadersListBox.SelectedItem);
        UserNameInput.Text = reader.Name;
        UserPassInput.Text = reader.Pass;
        IsAdminCheckBox.Checked = reader.IsAdmin;

        ShowUserBooks(reader);
    }
}
public void BooksListBox_SelectedIndexChanged(object sender, EventArgs e)
{
    if (BooksListBox.SelectedItem != null)
    {
        Book book = ((Book)BooksListBox.SelectedItem);
        BookTitleInput.Text = book.Title;
        BookAuthorInput.Text = book.Author;

        ShowBookReader(book);
    }
}

public void AuthorsListBox_SelectedIndexChanged(object sender, EventArgs e)
{
    if (AuthorsListBox.SelectedItem != null)
    {
        Author author = ((Author)AuthorsListBox.SelectedItem);
        AuthorNameInput.Text = author.Name;

        //BookAuthorNameInput.Text = author.Name;
    }
}
//Демонтрація книг які читає користувач
private void ShowUserBooks (Reader reader)
{
    Console.WriteLine(reader.Name);
    List<Book> userBookList = new List<Book>();
```

```
    foreach(Guid id in reader.Books_Ids)
    {
        Console.WriteLine(id);
        userBookList.Add(lib.GetBookById(id));
    }
    UserBooksLiistBox.DataSource = userBookList.ToList();
}
//Демонстрація користувача який читає книгу
private void ShowBookReader(Book book)
{
    if (book.IsOnUse == true)
    {
        List<Reader> bookReaders = new List<Reader>();
        bookReaders.Add(lib.GetUserById(book.Reader_ID));

        BookReaderListBox.DataSource = bookReaders.ToList();
    }
    else
    {
        List<string> resString = new List<string> { "Книга вільна" };
        BookReaderListBox.DataSource = resString;
    }
}









private void AdminForm_Load(object sender, EventArgs e)
{

}

private void button1_Click(object sender, EventArgs e)
```

```
{
    CreateComboBoxes(int.Parse(AuthorsCountInput.Text));
}

private void CreateComboBoxes(int count)
{
    List<ComboBox> comboBoxes = new List<ComboBox>();
    for (int i = 0; i < count; i++)
    {
        Console.WriteLine("Combox");
        ComboBox combox = CreateComboBox(i);
        this.Controls.Add(combox);
        comboBoxes.Add(combox);
    }
}

public ComboBox CreateComboBox(int i)
{
    ComboBox combox = new ComboBox();
    combox.Location = new System.Drawing.Point(25, 342 + (i * 30));
    combox.DataSource = lib.AuthorsList.ToList();
    combox.SelectedIndexChanged += Combox_SelectedIndexChanged;
    return combox;
}

public void Combox_SelectedIndexChanged(object sender, EventArgs e)
{
    label11.Text = "china";
}

public void LogOutBtn_Click(object sender, EventArgs e)
{
    AuthForm authForm = new AuthForm();
    authForm.Show();
    this.Close();
}
```

```
        public void AddAuthorBtn_Click_1(object sender, EventArgs e)
        {
            Console.WriteLine("ADD BOOK");
            string name = AuthorNameInput.Text;
            lib.AddNewAuthor(name);
            RefreshAuthorsListBox();
        }
    }
}
```

2) UserForm.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace final_project
{
    public partial class UserForm : Form
    {
        Library lib = new Library();

        Reader user;
        public UserForm(Guid id)
        {
            InitializeComponent();
            Console.WriteLine(id);
            user = lib.GetUserById(id);
            UserNameL.Text = user.Name;
            RefreshBooksList(0);
        }

        private void RefreshBooksList(int type) // 0-показати всі книги, 1-показати тільки книги користувача
        {
            if(type == 0)
```

```
    {
        BooksListBox.DataSource = lib.BooksList.ToList();
        UserPassL.Text = "All books";
    }
    else if(type == 1)
    {
        List<Book> Userbooks = new List<Book>();
        Console.WriteLine("CHINAAAAAA");
        foreach(Guid id in user.Books_Ids)
        {
            Console.WriteLine(id);
            Userbooks.Add(lib.BooksList.Find(x => x._Id == id));
        }
        UserPassL.Text = "My books";
        BooksListBox.DataSource = Userbooks;
    }

}
private void ShowMyBooks_Click(object sender, EventArgs e)
{
    RefreshBooksList(1);
}
private void ShowAllAvailableBooks_Click(object sender, EventArgs e)
{
    RefreshBooksList(0);
}

private void BorrowBook_Click(object sender, EventArgs e)
{
    if (BooksListBox.SelectedItem != null)
    {
        Book book = ((Book)BooksListBox.SelectedItem);
        lib.GiveBookToUser(book, user);
    }
}
```

```
        private void ReturnBook_Click(object sender, EventArgs e)
        {
            if (BooksListBox.SelectedItem != null)
            {
                Book book = ((Book)BooksListBox.SelectedItem);
                lib.ReturnBookFromUser(book, user);
            }
            RefreshBooksList(1);
        }

        private void SerchBtn_Click(object sender, EventArgs e)
        {
            string searchString = SearchInput.Text;
            List<Book> searchResults = lib.BooksList.FindAll(x => x.Title.Contains(searchString));
            BooksListBox.DataSource = searchResults;
        }

        private void LogOutBtn_Click(object sender, EventArgs e)
        {
            AuthForm authForm = new AuthForm();
            authForm.Show();
            this.Close();
        }
    }
}
```

3) AuthForm.cs

```
using System;
using System.Windows.Forms;

namespace final_project
{
    public partial class AuthForm : Form
    {
        Library lib = new Library();
        public AuthForm()
        {
```

```
    InitializeComponent();
}

private void SignInBtn_Click(object sender, EventArgs e)
{
    string userName = UserNameInput.Text;
    string userPass = UserPassInput.Text;

    Reader reader = lib.SignIn(userName, userPass);
    if (reader == null)
    {
        resLabel.Text = "Невірний логін або пароль";
    }
    else if(reader.IsAdmin == true) // Перевірка чи коримстувач адмін
    {
        OpenAdminForm();
    }
    else
    {
        OpenUserForm(reader._Id);
    }
}

private void RegBtn_Click(object sender, EventArgs e)
{
    string userName = UserNameInput.Text;
    string userPass = UserPassInput.Text;
    Guid id = lib.AddNewUser(userName, userPass, false);
    OpenUserForm(id);
}

private void OpenUserForm(Guid id)
{
    UserForm userForm = new UserForm(id);
    userForm.Show();
}
```

```
        private void OpenAdminForm()
        {
            AdminForm adminForm = new AdminForm();
            adminForm.Show();
        }
    }
}
```