



**WYŻSZA SZKOŁA  
INFORMATYKI i ZARZĄDZANIA**  
z siedzibą w Rzeszowie

## **KOLEGIUM INFORMATYKI STOSOWANEJ**

**Kierunek: INFORMATYKA**

**Specjalność: Inżynieria danych**

Mateusz Jocek

Nr albumu studenta: w68378

### ***System zarządzania biblioteką – aplikacja konsolowa w języku C#***

Prowadzący: mgr inż. Ewa Żesławska

**Projekt z przedmiotu „Programowanie obiektowe”**

**Rzeszów 2025**



# Spis treści

<b>Wstęp</b>	<b>4</b>
<b>1 Opis założeń projektu</b>	<b>5</b>
1.1 Cele projektu . . . . .	5
1.2 Wymagania funkcjonalne i нефункционалне . . . . .	5
1.2.1 Wymagania funkcjonalne . . . . .	5
1.2.2 Wymagania нефункционалне . . . . .	6
<b>2 Minimalne wymagania sprzętowe, system operacyjny, база данных i структура klas</b>	<b>7</b>
2.1 Minimalne wymagania sprzętowe i system operacyjny . . . . .	7
2.2 Opis bazy danych . . . . .	7
2.3 Struktura klas . . . . .	7
2.3.1 Hierarchia klas . . . . .	7
2.3.2 Klasy wspierające . . . . .	8
2.4 Diagram klas . . . . .	9
2.5 Podsumowanie . . . . .	10
<b>3 Harmonogram realizacji projektu</b>	<b>11</b>
3.1 Wprowadzenie . . . . .	11
3.2 Etapy realizacji projektu . . . . .	11
3.3 Diagram Gantta . . . . .	11
3.4 Szczegółowy harmonogram . . . . .	11
3.5 Podsumowanie . . . . .	12
<b>4 Prezentacja warstwy użytkowej projektu</b>	<b>13</b>
4.1 Wstęp . . . . .	13
4.2 Strona startowa . . . . .	14
4.3 Dodawanie książek . . . . .	17
4.4 Edycja informacji o książce . . . . .	18
4.5 Usuwanie książek . . . . .	19
4.6 Wyświetlanie listy książek . . . . .	20
4.7 Dodawanie użytkowników . . . . .	21
4.8 Wyświetlanie listy użytkowników . . . . .	22
4.9 Wypożyczanie książek . . . . .	23
4.10 Zwrot książki . . . . .	24
4.11 Podsumowanie . . . . .	25
<b>Spis rysunków</b>	<b>27</b>
<b>Spis tablic</b>	<b>28</b>

# Wstęp

Biblioteki odgrywają istotną rolę w przechowywaniu i udostępnianiu wiedzy. Wraz z postępem technologicznym coraz więcej instytucji korzysta z systemów informatycznych do automatyzacji procesów zarządzania zbiorami oraz obsługi użytkowników. Wprowadzenie takich rozwiązań pozwala na oszczędność czasu, zwiększenie efektywności oraz poprawę jakości świadczonych usług.

Celem niniejszego projektu jest opracowanie aplikacji konsolowej umożliwiającej zarządzanie biblioteką. Aplikacja pozwoli na przechowywanie informacji o książkach, użytkownikach oraz operacjach wypożyczeń i zwrotów. Projekt został zrealizowany w oparciu o platformę .NET w wersji 6, co zapewnia stabilność oraz długoterminowe wsparcie techniczne (LTS). Wybór tej technologii umożliwia wykorzystanie nowoczesnych funkcji języka C# oraz zapewnia zgodność z różnymi środowiskami systemowymi.

System jest oparty na zasadach programowania obiektowego, co umożliwia lepszą organizację kodu oraz łatwiejsze rozbudowywanie aplikacji w przyszłości. Modułowa architektura pozwala na wprowadzanie nowych funkcjonalności bez zakłócania działania istniejących komponentów.

Projekt realizowany jest w ramach przedmiotu „Programowanie obiektowe” i ma na celu praktyczne zastosowanie zasad projektowania i implementacji systemów informatycznych. Dane w aplikacji będą przechowywane w plikach tekstowych, co zapewni prostotę użytkowania i zgodność z minimalnymi wymaganiami sprzętowymi. System ten będzie przyjazny dla użytkownika i intuicyjny w obsłudze, a jego funkcjonalność może być rozwijana w kolejnych etapach prac.

# Rozdział 1

## Opis założeń projektu

### 1.1 Cele projektu

Celem projektu jest stworzenie aplikacji konsolowej wspierającej procesy zarządzania biblioteką. System umożliwia przechowywanie informacji o książkach, użytkownikach oraz operacjach wypożyczeń i zwrotów. Realizacja projektu odbywa się w ramach przedmiotu „Programowanie obiektowe” i ma na celu praktyczne zastosowanie zasad projektowania oraz implementacji systemów informatycznych z wykorzystaniem programowania obiektowego.

Kluczowym problemem, który rozwiązuje system, jest usprawnienie organizacji i zarządzania zasobami biblioteki. Tradycyjne metody, takie jak papierowe rejestry czy arkusze kalkulacyjne, bywają czasochłonne i podatne na błędy, co wpływa negatywnie na efektywność pracy. Proponowany system ma na celu zminimalizowanie tych trudności poprzez:

- Zautomatyzowanie procesów operacyjnych, takich jak wypożyczenia i zwroty,
- Umożliwienie szybkiego dostępu do danych o książkach i użytkownikach,
- Zwiększenie przejrzystości zarządzania zbiorami bibliotecznymi.

Projekt zakłada stworzenie aplikacji konsolowej o modularnej architekturze, co pozwoli na jej łatwą rozbudowę w przyszłości. Wykorzystanie języka C# oraz platformy .NET 6 umożliwia implementację nowoczesnych rozwiązań technologicznych, a także zapewnia zgodność aplikacji z różnymi środowiskami systemowymi.

### 1.2 Wymagania funkcjonalne i нефункционалне

#### 1.2.1 Wymagania funkcjonalne

Aplikacja zarządzania biblioteką realizuje następujące funkcjonalności:

- **Dodawanie, edytowanie i usuwanie informacji o książkach:** Administratorzy mogą wprowadzać nowe pozycje do bazy danych, aktualizować dane istniejących książek oraz usuwać je w przypadku wycofania z obiegu.
- **Zarządzanie użytkownikami:** System umożliwia dodawanie, modyfikację i usuwanie użytkowników z bazy, zapewniając odpowiedni poziom autoryzacji dla każdego profilu.
- **Rejestrowanie wypożyczeń i zwrotów:** Każda operacja związana z wypożyczeniem lub zwrotem książki jest rejestrowana z uwzględnieniem identyfikatora użytkownika oraz szczegółowych informacji o książce.
- **Wyszukiwanie danych:** System umożliwia szybkie wyszukiwanie książek i użytkowników według takich kryteriów jak tytuł, autor, numer ID czy dane użytkownika.

## 1.2.2 Wymagania niefunkcjonalne

System spełnia także następujące wymagania niefunkcjonalne:

- **Wydażność:** System obsługuje do 1000 rekordów bez zauważalnych opóźnień.
- **Bezpieczeństwo:** Aplikacja chroni dane użytkowników oraz historię wypożyczeń przed nieautoryzowanym dostępem. Obsługiwane jest logowanie z hasłami oraz podstawowe mechanizmy autoryzacji.
- **Przenośność:** Aplikacja może być uruchamiana na systemach Windows z zainstalowanym .NET Framework lub .NET Core.
- **Rozszerzalność:** Architektura systemu pozwala na integrację z zewnętrznymi bazami danych oraz rozbudowę o dodatkowe funkcjonalności w przyszłości.
- **Użyteczność:** Interfejs użytkownika jest prosty i intuicyjny, co minimalizuje potrzebę dodatkowego szkolenia.

System został zaprojektowany tak, aby sprostać wymaganiom zarówno użytkowników końcowych, jak i administratorów, zapewniając jednocześnie zgodność z minimalnymi wymaganiami sprzętowymi.

# Rozdział 2

## Minimalne wymagania sprzętowe, system operacyjny, baza danych i struktura klas

### 2.1 Minimalne wymagania sprzętowe i system operacyjny

System zarządzania biblioteką został zaprojektowany tak, aby mógł działać na szerokiej gamie urządzeń o minimalnej specyfikacji. Wymagania sprzętowe są następujące:

- Procesor: 2 GHz, dwurdzeniowy,
- Pamięć RAM: 4 GB,
- Wolne miejsce na dysku: 100 MB,
- Monitor o rozdzielczości co najmniej 1024x768 pikseli.

System operacyjny:

- Windows 10 lub nowszy,
- Wsparcie dla platformy .NET 6 lub .NET Core.

### 2.2 Opis bazy danych

Dane w systemie są przechowywane w formacie plików tekstowych CSV, co pozwala na łatwe ich edytowanie i przeglądanie.

### 2.3 Struktura klas

Aplikacja została zaprojektowana zgodnie z zasadami programowania obiektowego. W systemie wykorzystano siedem klas, z których pięć tworzy hierarchię dziedziczenia.

#### 2.3.1 Hierarchia klas

- **BaseEntity**: Klasa bazowa zawierająca wspólne atrybuty:
  - `int ID` – unikalny identyfikator,
  - `DateTime CreatedAt` – data utworzenia,
  - `DateTime UpdatedAt` – data ostatniej modyfikacji.
- **Book (dziedziczy po BaseEntity)**: Reprezentuje książkę w systemie:

- `string Title` – tytuł książki,
- `string Author` – autor książki,
- `int Year` – rok wydania,
- `bool IsAvailable` – dostępność książki.

- **User (dziedziczy po BaseEntity):** Przechowuje dane użytkownika:

- `string Name` – imię użytkownika,
- `string Surname` – nazwisko,
- `string PhoneNumber` – numer telefonu.

### 2.3.2 Klasy wspierające

System zawiera także klasy wspierające, które odpowiadają za zarządzanie danymi oraz operacje wejścia/wyjścia.

- **LibraryManager:** Klasa zarządzająca operacjami na książkach, użytkownikach i transakcjach. Obsługuje operacje CRUD.

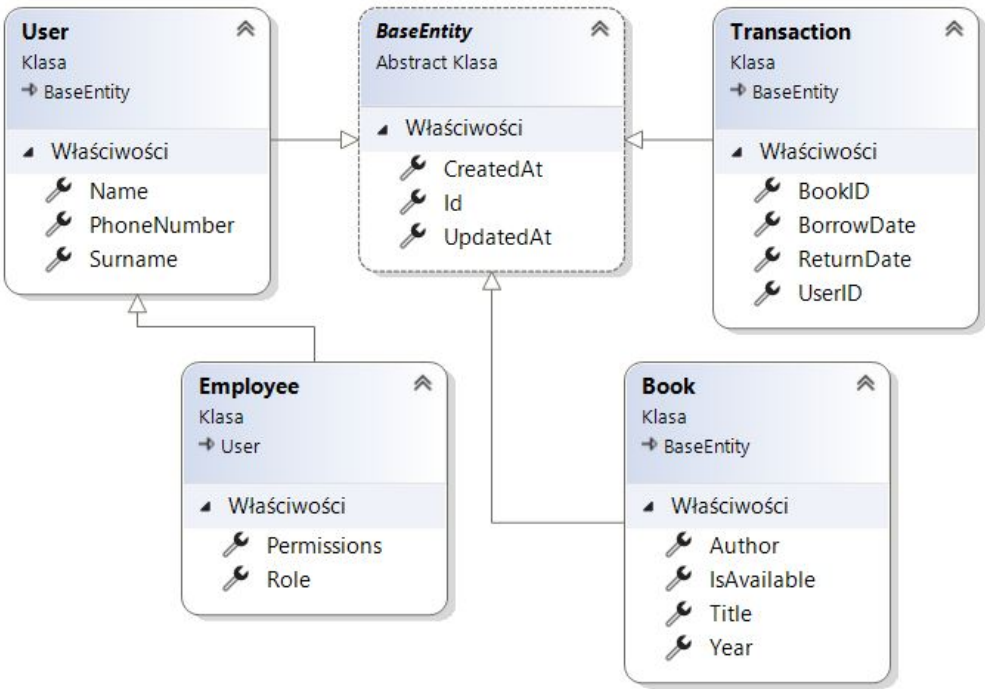
- `void AddBook(Book book)` – dodaje nową książkę do systemu.
- `void RemoveBook(int bookId)` – usuwa książkę na podstawie jej identyfikatora.
- `void RegisterUser(User user)` – rejestruje nowego użytkownika.
- `void ProcessTransaction(Transaction transaction)` – obsługuje proces wypożyczenia i zwrotu książek.
- `List<Book> SearchBooks(string keyword)` – umożliwia wyszukiwanie książek według tytułu lub autora.

- **FileHandler:** Klasa obsługująca operacje odczytu i zapisu danych w plikach CSV.

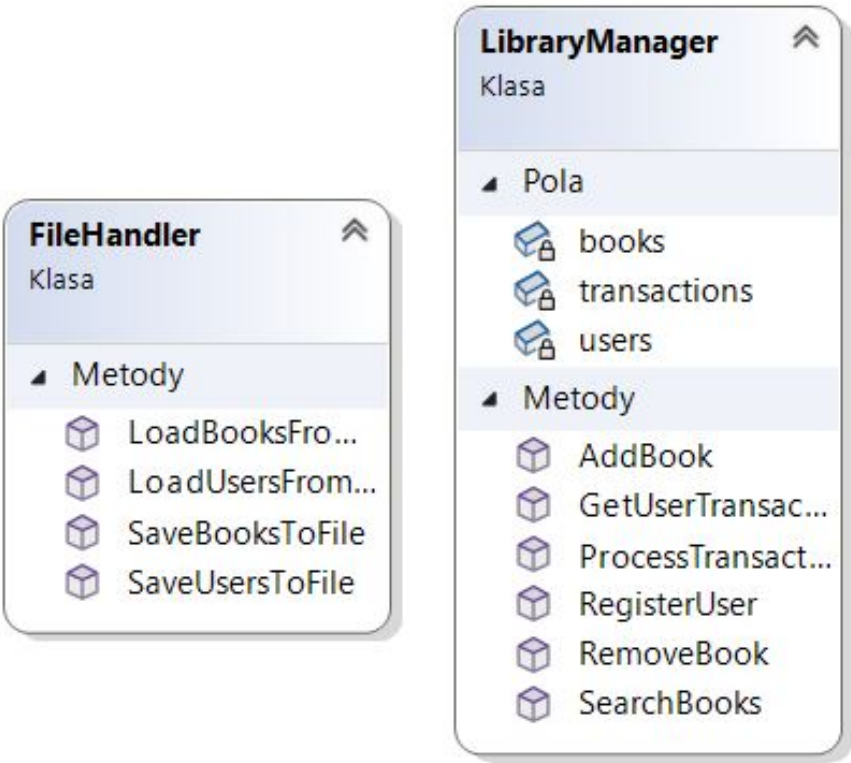
- `void SaveBooksToFile(List<Book> books, string filePath)` – zapisuje listę książek do pliku CSV.
- `List<Book> LoadBooksFromFile(string filePath)` – wczytuje listę książek z pliku CSV.
- `void SaveUsersToFile(List<User> users, string filePath)` – zapisuje listę użytkowników do pliku.



## 2.4 Diagram klas



Rysunek 2.1: Diagram klas – hierarchia dziedziczenia



Rysunek 2.2: Diagram klas – klasy wspierające i ich metody

## 2.5 Podsumowanie

Rozdział przedstawia minimalne wymagania sprzętowe, strukturę bazy danych oraz opisuje hierarchię klas w systemie. Dzięki modularnej budowie systemu możliwe jest łatwe rozszerzanie jego funkcjonalności oraz integracja z bardziej zaawansowanymi rozwiązaniami w przyszłości.

# Rozdział 3

## Harmonogram realizacji projektu

### 3.1 Wprowadzenie

Harmonogram realizacji projektu został przedstawiony w postaci diagramu Gantta. Diagram ten wizualizuje planowany czas trwania poszczególnych etapów projektu oraz ich kolejność.

### 3.2 Etapy realizacji projektu

Projekt realizowany jest w kilku etapach:

- **Przygotowanie koncepcji** – analiza wymagań, opracowanie głównych założeń projektu.
- **Projektowanie systemu** – tworzenie modelu systemu, struktury baz danych oraz diagramów UML, w tym diagramu klas i przypadków użycia, które wizualizują architekturę oraz interakcje użytkownika z systemem.
- **Implementacja funkcji** – programowanie kluczowych funkcjonalności systemu.
- **Testowanie i poprawki** – weryfikacja działania aplikacji, eliminacja błędów.
- **Finalizacja i obrona** – dokumentacja i prezentacja projektu.

### 3.3 Diagram Gantta

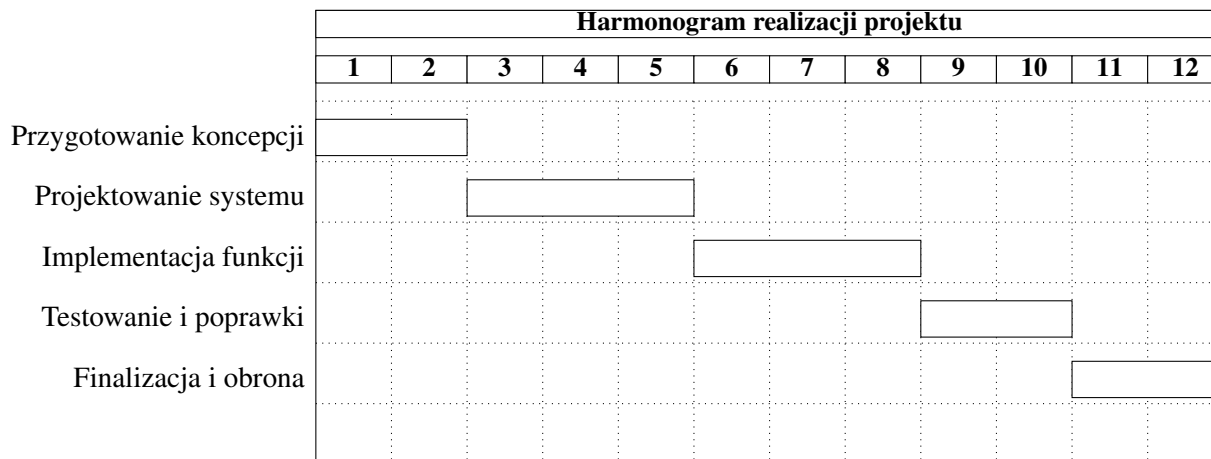
Poniżej przedstawiono harmonogram realizacji projektu w postaci diagramu Gantta, który prezentuje czas trwania poszczególnych etapów.

### 3.4 Szczegółowy harmonogram

Diagram Gantta można także przedstawić w postaci tabeli, która zawiera dokładne daty rozpoczęcia i zakończenia każdego etapu:

Nr	Etap	Rozpoczęcie	Zakończenie
1	Przygotowanie koncepcji i dokumentacji	01.12.2024	30.12.2024
2	Projektowanie systemu	01.01.2025	15.01.2025
3	Implementacja podstawowych funkcjonalności	16.01.2025	05.02.2025
4	Rozbudowa systemu i testowanie	06.02.2025	15.02.2025
5	Finalizacja dokumentacji i repozytorium	16.02.2025	18.02.2025
6	Obrona projektu	20.02.2025	20.02.2025

Tabela 3.1: Harmonogram realizacji projektu – podział na etapy



Rysunek 3.1: Diagram Gantta – harmonogram realizacji projektu

## 3.5 Podsumowanie

Diagram Gantta przedstawia szczegółowy plan realizacji projektu, dzieląc go na poszczególne etapy oraz określając ich czas trwania. Dzięki temu można lepiej kontrolować postęp prac i przewidywać potencjalne opóźnienia.

Wszystkie pliki związane z projektem, w tym kod źródłowy, dokumentacja oraz diagramy, będą dostępne w repozytorium GitHub pod adresem: [https://github.com/w68378/Programowanie\\_obiektowe\\_laboratorium/tree/main/Projekt/ConsoleAppl](https://github.com/w68378/Programowanie_obiektowe_laboratorium/tree/main/Projekt/ConsoleAppl).

# Rozdział 4

## Prezentacja warstwy użytkowej projektu

### 4.1 Wstęp

W niniejszym rozdziale zostanie zaprezentowana warstwa użytkowa aplikacji do zarządzania biblioteką. System umożliwia wykonywanie różnych operacji związanych z zarządzaniem książkami, użytkownikami oraz historią wypożyczeń. Każda funkcjonalność zostanie szczegółowo opisana wraz ze zrzutami ekranu przedstawiającymi działanie systemu oraz wizualizacjami interakcji użytkownika.

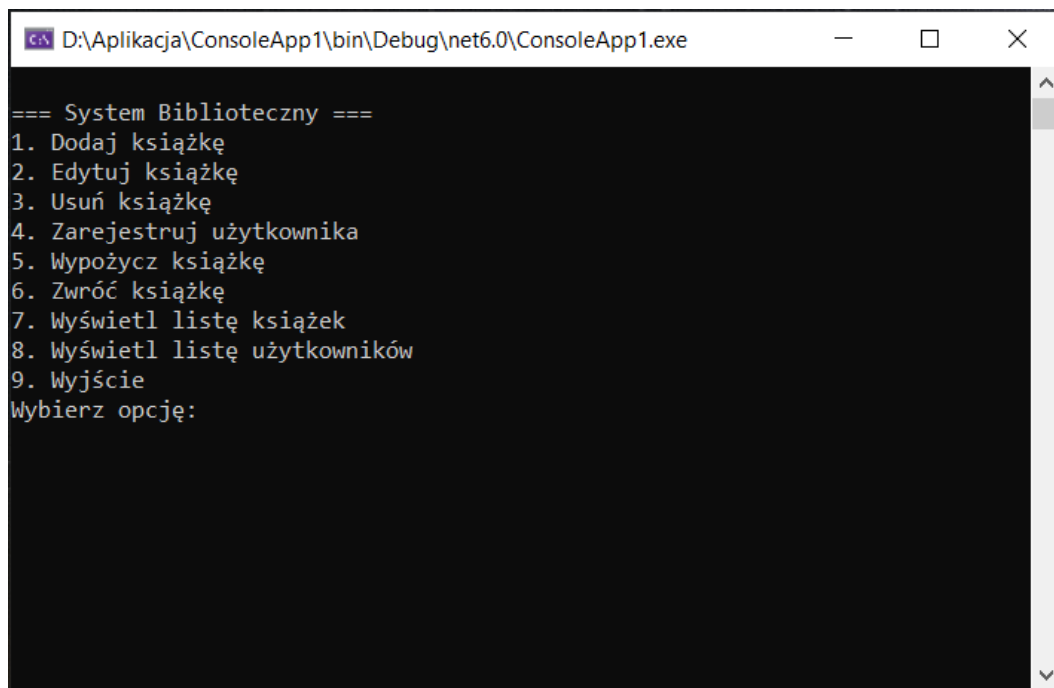
Poniżej przedstawiono pełną listę dostępnych funkcjonalności systemu:

- **Strona startowa** – prezentacja ekranu początkowego systemu.
- **Dodawanie książek** – użytkownik może dodać nową książkę do katalogu biblioteki.
- **Edycja informacji o książce** – użytkownik może zmienić dane dotyczące istniejącej książki.
- **Usuwanie książek** – użytkownik może usunąć książkę z systemu.
- **Wyświetlanie listy książek** – użytkownik może zobaczyć wszystkie książki dostępne w bibliotece.
- **Dodawanie użytkowników** – użytkownik może zarejestrować nowego użytkownika biblioteki.
- **Wyświetlanie listy użytkowników** – użytkownik może zobaczyć wszystkich użytkowników biblioteki.
- **Wypożyczanie książek** – użytkownik może wypożyczyć książkę, jeśli jest dostępna.
- **Zwrot książki** – użytkownik może zwrócić wcześniej wypożyczoną książkę.

Każda z powyższych funkcji zostanie dokładnie opisana w kolejnych sekcjach, wraz z odpowiednimi wizualizacjami przedstawiającymi interakcję użytkownika z systemem.

## 4.2 Strona startowa

Strona startowa aplikacji to ekran powitalny, który pozwala użytkownikowi na wybór dostępnych opcji zarządzania systemem bibliotecznym.



Rysunek 4.1: Strona startowa aplikacji

```
static void Main()
{
    // Tworzenie obiektu LibraryManager do zarządzania biblioteką
    LibraryManager libraryManager = new LibraryManager();

    // Pętla nieskończona - program działa dopóki użytkownik nie wybierze opcji wyjścia
    while (true)
    {
        // Wyświetlenie menu głównego aplikacji
        Console.WriteLine("\n=== System Biblioteczny ===");
        Console.WriteLine("1. Dodaj książkę");
        Console.WriteLine("2. Edytuj książkę");
        Console.WriteLine("3. Usuń książkę");
        Console.WriteLine("4. Zarejestruj użytkownika");
        Console.WriteLine("5. Wypożycz książkę");
        Console.WriteLine("6. Zwróć książkę");
        Console.WriteLine("7. Wyświetl listę książek");
        Console.WriteLine("8. Wyświetl listę użytkowników");
        Console.WriteLine("9. Wyjście");
        Console.Write("Wybierz opcję: ");
    }
}
```

Rysunek 4.2: Kod źródłowy obsługi menu głównego

```

// Odczyt wyboru użytkownika i jego konwersja na liczbę
if (!int.TryParse(Console.ReadLine(), out int choice)) continue;

// Obsługa wyboru użytkownika poprzez instrukcję switch
switch (choice)
{
    case 1:
        // Dodawanie nowej książki
        Console.Write("Tytuł: ");
        string title = Console.ReadLine() ?? "";
        Console.Write("Autor: ");
        string author = Console.ReadLine() ?? "";
        Console.Write("Rok wydania: ");
        if (!int.TryParse(Console.ReadLine(), out int year)) continue;

        // Przekazanie danych do obiektu LibraryManager
        libraryManager.AddBook(new Book { Title = title, Author = author, Year = year });
        break;

    case 2:
        // Edycja informacji o książce
        libraryManager.ShowBooks(); // Wyświetlenie listy książek
        Console.Write("Podaj ID książki do edycji: ");
        if (!int.TryParse(Console.ReadLine(), out int editId)) continue;

        Console.Write("Nowy tytuł: ");
        string newTitle = Console.ReadLine() ?? "";
        Console.Write("Nowy autor: ");
        string newAuthor = Console.ReadLine() ?? "";
        Console.Write("Nowy rok wydania: ");
        if (!int.TryParse(Console.ReadLine(), out int newYear)) continue;

        // Przekazanie zmienionych danych do metody edycji książki
        libraryManager.EditBook(editId, newTitle, newAuthor, newYear);
        break;

    case 3:
        // Usuwanie książki
        libraryManager.ShowBooks(); // Wyświetlenie listy książek
        Console.Write("Podaj ID książki do usunięcia: ");
        if (!int.TryParse(Console.ReadLine(), out int removeId)) continue;
        libraryManager.RemoveBook(removeId);
        break;
}

```

Rysunek 4.3: Kod źródłowy obsługi menu głównego

```

case 4:
    // Rejestracja nowego użytkownika
    Console.WriteLine("Imię: ");
    string firstName = Console.ReadLine() ?? "";
    Console.WriteLine("Nazwisko: ");
    string lastName = Console.ReadLine() ?? "";
    Console.WriteLine("Numer telefonu: ");
    string phone = Console.ReadLine() ?? "";

    // Przekazanie danych do metody rejestracji użytkownika
    libraryManager.RegisterUser(new User { Name = firstName, Surname = lastName, PhoneNumber = phone });
    break;

case 5:
    // Wypożyczanie książki
    libraryManager.ShowBooks(); // Wyświetlenie listy książek
    Console.WriteLine("Podaj ID książki do wypożyczenia: ");
    if (!int.TryParse(Console.ReadLine(), out int bookId)) continue;
    Console.WriteLine("Podaj ID użytkownika: ");
    if (!int.TryParse(Console.ReadLine(), out int userId)) continue;

    // Wywołanie metody wypożyczenia książki
    libraryManager.BorrowBook(bookId, userId);
    break;

case 6:
    // Zwrot książki
    Console.WriteLine("Podaj ID książki do zwrotu: ");
    if (!int.TryParse(Console.ReadLine(), out int returnBookId)) continue;
    Console.WriteLine("Podaj ID użytkownika: ");
    if (!int.TryParse(Console.ReadLine(), out int returnUserId)) continue;

    // Wywołanie metody zwrotu książki
    libraryManager.ReturnBook(returnBookId, returnUserId);
    break;

case 7:
    // Wyświetlenie listy książek
    libraryManager.ShowBooks();
    break;

case 8:
    // Wyświetlenie listy użytkowników
    libraryManager.ShowUsers();
    break;

case 9:
    // Wyjście z programu
    return;

default:
    // Obsługa nieprawidłowego wyboru
    Console.WriteLine("Nieprawidłowy wybór.");
    break;
}
}
}
}

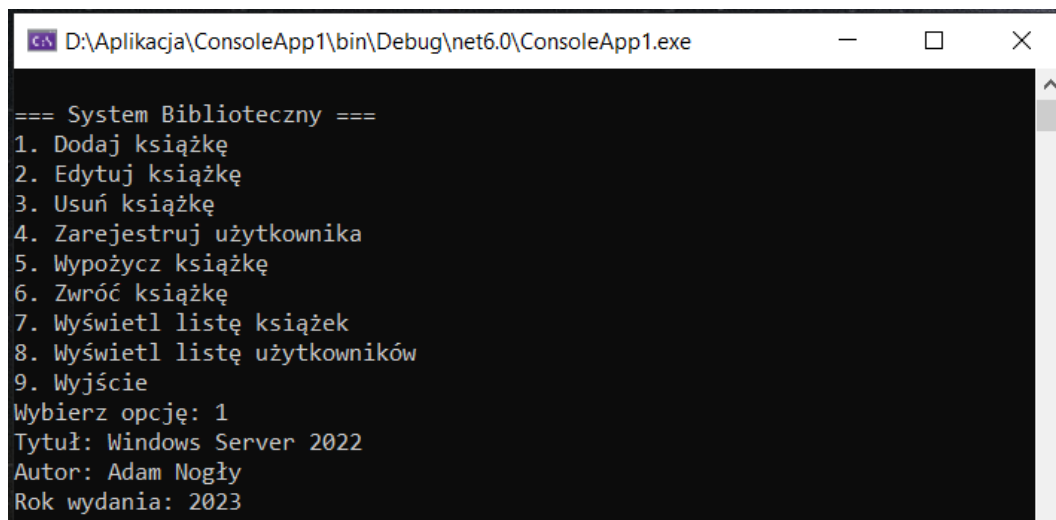
```

Rysunek 4.4: Kod źródłowy obsługi menu głównego



## 4.3 Dodawanie książek

Użytkownik może dodać nową książkę do katalogu biblioteki. Aplikacja wymaga podania tytułu, autora i roku wydania.



Rysunek 4.5: Dodawanie nowej książki

```
// Dodawanie nowej książki do systemu
1 odwołanie
public void AddBook(Book book)
{
    if (book != null) // Sprawdzenie, czy book nie jest null
    {
        books.Add(book);
        fileHandler.SaveBooksToFile(books, "books.csv");
    }
}
```

Rysunek 4.6: Kod źródłowy dodawania książki

## 4.4 Edycja informacji o książce

Użytkownik ma możliwość edycji danych książki, jeśli wymaga ona aktualizacji.

```
cs D:\Aplikacja\ConsoleApp1\bin\Debug\net6.0\ConsoleApp1.exe

=== System Biblioteczny ===
1. Dodaj książkę
2. Edytuj książkę
3. Usuń książkę
4. Zarejestruj użytkownika
5. Wypożycz książkę
6. Zwróć książkę
7. Wyświetl listę książek
8. Wyświetl listę użytkowników
9. Wyjście
Wybierz opcję: 2
ID: 1, Tytuł: Informatyka śledcza, Autor: Tomasz Kaniecki, Rok: 2000, Dostępność: True
ID: 0, Tytuł: Windows Server 2022, Autor: Adam Nogły, Rok: 2023, Dostępność: True
Podaj ID książki do edycji: 1
Nowy tytuł: Informatyka
Nowy autor: Tomasz Kaniecki
Nowy rok wydania: 2000
```

Rysunek 4.7: Edycja informacji o książce

```
// Edytowanie istniejącej książki na podstawie ID
1 odwołanie
public void EditBook(int bookId, string newTitle, string newAuthor, int newYear)
{
    Book? book = books.FirstOrDefault(b => b.ID == bookId); // '?' pozwala na null
    if (book != null)
    {
        book.Title = newTitle;
        book.Author = newAuthor;
        book.Year = newYear;
        fileHandler.SaveBooksToFile(books, "books.csv");
    }
    else
    {
        Console.WriteLine("Nie znaleziono książki o podanym ID.");
    }
}
```

Rysunek 4.8: Kod źródłowy edycji książki

## 4.5 Usuwanie książek

System umożliwia użytkownikowi usunięcie książki z katalogu bibliotecznego.



```
D:\Aplikacja\ConsoleApp1\bin\Debug\net6.0\ConsoleApp1.exe

=== System Biblioteczny ===
1. Dodaj książkę
2. Edytuj książkę
3. Usuń książkę
4. Zarejestruj użytkownika
5. Wypożycz książkę
6. Zwróć książkę
7. Wyświetl listę książek
8. Wyświetl listę użytkowników
9. Wyjście
Wybierz opcję: 3
ID: 1, Tytuł: Informatyka, Autor: Tomasz Kaniecki, Rok: 2000, Dostępność: True
ID: 0, Tytuł: Windows Server 2022, Autor: Adam Nogły, Rok: 2023, Dostępność: True
Podaj ID książki do usunięcia: 1
```

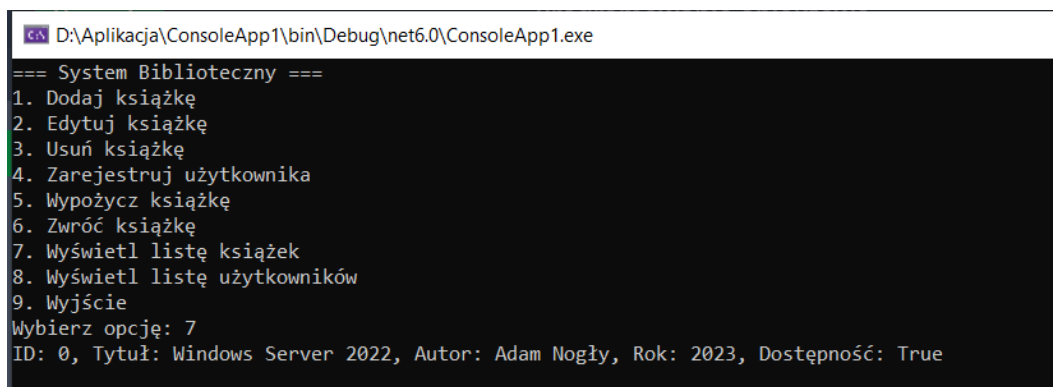
Rysunek 4.9: Usuwanie książki

```
// Usuwanie książki z katalogu bibliotecznego
1 odwołanie
public void RemoveBook(int bookId)
{
    Book? book = books.FirstOrDefault(b => b.ID == bookId);
    if (book != null)
    {
        books.Remove(book);
        fileHandler.SaveBooksToFile(books, "books.csv");
    }
    else
    {
        Console.WriteLine("Nie znaleziono książki o podanym ID.");
    }
}
```

Rysunek 4.10: Kod źródłowy usuwania książki

## 4.6 Wyświetlanie listy książek

System umożliwia użytkownikowi przeglądanie katalogu książek dostępnych w bibliotece.



```
D:\Aplikacja\ConsoleApp1\bin\Debug\net6.0\ConsoleApp1.exe

=== System Biblioteczny ===
1. Dodaj książkę
2. Edytuj książkę
3. Usuń książkę
4. Zarejestruj użytkownika
5. Wypożycz książkę
6. Zwróć książkę
7. Wyświetl listę książek
8. Wyświetl listę użytkowników
9. Wyjście
Wybierz opcję: 7
ID: 0, Tytuł: Windows Server 2022, Autor: Adam Nogły, Rok: 2023, Dostępność: True
```

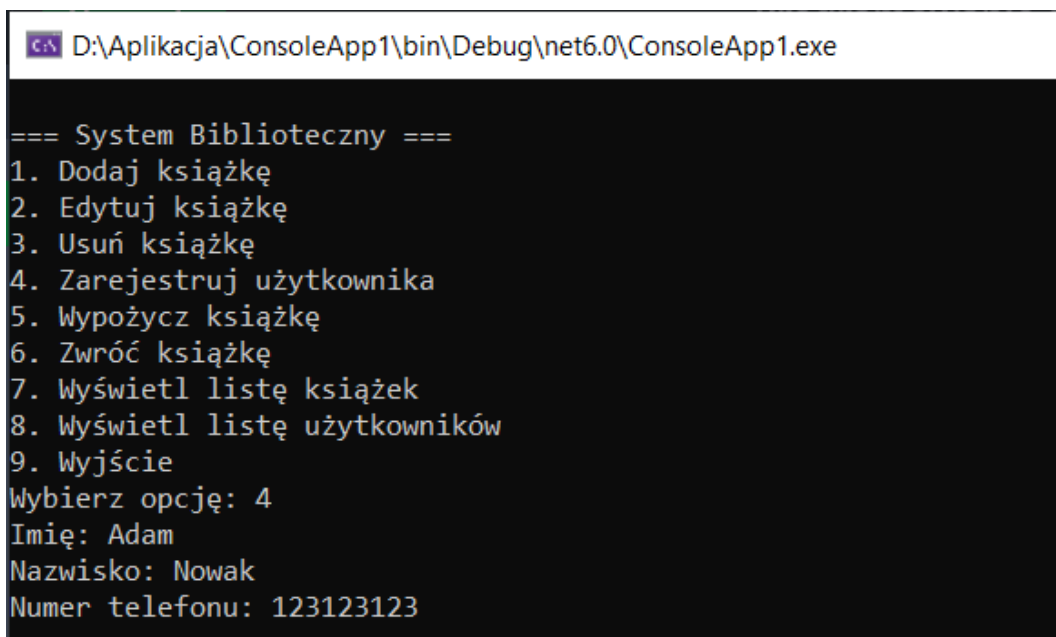
Rysunek 4.11: Lista książek w bibliotece

```
// Wyświetlanie listy książek
Odwwołania: 4
public void ShowBooks()
{
    if (books != null && books.Count > 0) // Sprawdzenie, czy lista nie jest pusta
    {
        foreach (var book in books)
        {
            Console.WriteLine($"ID: {book.ID}, Tytuł: {book.Title}, Autor: {book.Author}, Rok: {book.Year}, " +
                               $"Dostępność: {book.IsAvailable}");
        }
    }
    else
    {
        Console.WriteLine("Brak książek w systemie.");
    }
}
```

Rysunek 4.12: Kod źródłowy wyświetlania listy książek

## 4.7 Dodawanie użytkowników

Użytkownik może dodać nowego użytkownika biblioteki.



```
D:\Aplikacja\ConsoleApp1\bin\Debug\net6.0\ConsoleApp1.exe

=== System Biblioteczny ===
1. Dodaj książkę
2. Edytuj książkę
3. Usuń książkę
4. Zarejestruj użytkownika
5. Wypożycz książkę
6. Zwróć książkę
7. Wyświetl listę książek
8. Wyświetl listę użytkowników
9. Wyjście
Wybierz opcję: 4
Imię: Adam
Nazwisko: Nowak
Numer telefonu: 123123123
```

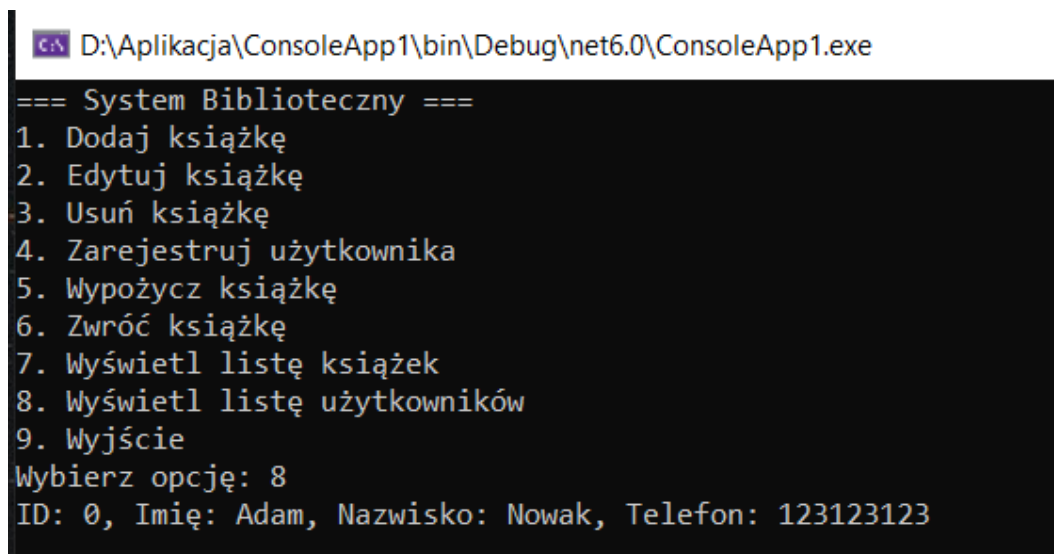
Rysunek 4.13: Dodawanie nowego użytkownika

```
// Rejestracja nowego użytkownika
1 odwołanie
public void RegisterUser(User user)
{
    if (user != null) // Sprawdzenie, czy user nie jest null
    {
        users.Add(user);
        fileHandler.SaveUsersToFile(users, "users.csv");
    }
}
```

Rysunek 4.14: Kod źródłowy dodawania użytkownika

## 4.8 Wyświetlanie listy użytkowników

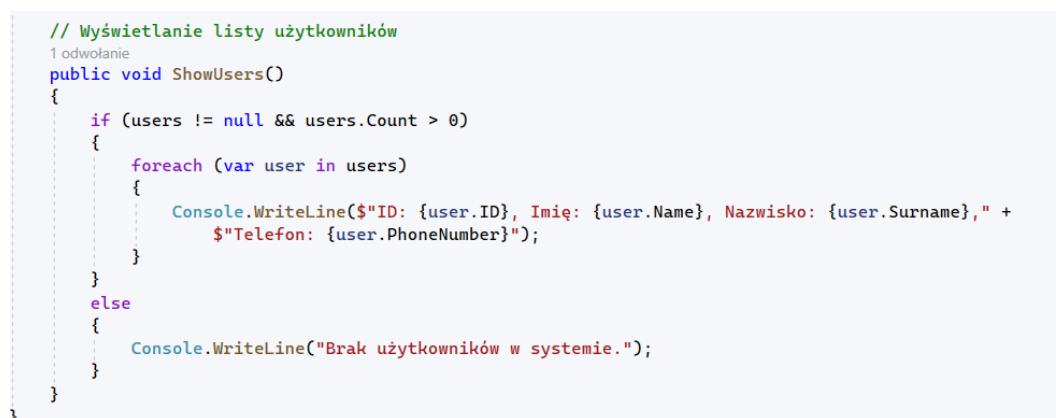
System umożliwia użytkownikowi przeglądanie listy użytkowników biblioteki.



The screenshot shows a console window titled "D:\Aplikacja\ConsoleApp1\bin\Debug\net6.0\ConsoleApp1.exe". The output displays a menu titled "=== System Biblioteczny ===" with nine options. Option 8, "Wyświetl listę użytkowników", is selected. Below the menu, the user list is displayed: "ID: 0, Imię: Adam, Nazwisko: Nowak, Telefon: 123123123".

```
D:\Aplikacja\ConsoleApp1\bin\Debug\net6.0\ConsoleApp1.exe
=== System Biblioteczny ===
1. Dodaj książkę
2. Edytuj książkę
3. Usuń książkę
4. Zarejestruj użytkownika
5. Wypożycz książkę
6. Zwróć książkę
7. Wyświetl listę książek
8. Wyświetl listę użytkowników
9. Wyjście
Wybierz opcję: 8
ID: 0, Imię: Adam, Nazwisko: Nowak, Telefon: 123123123
```

Rysunek 4.15: Lista użytkowników biblioteki



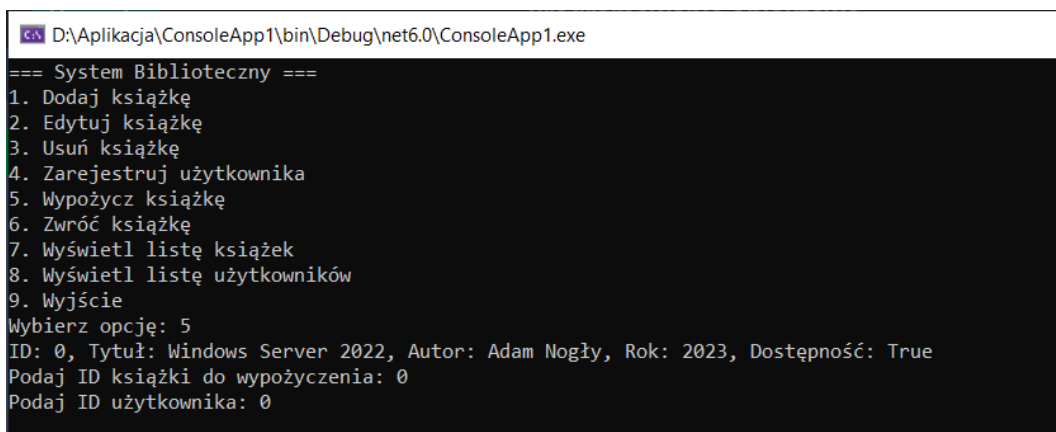
The screenshot shows the C# source code for the `ShowUsers()` method. It includes a comment in Polish: "`// Wyświetlanie listy użytkowników`" and "`1 odwołanie`". The code uses an `if` statement to check if the `users` list is not null and has more than zero elements. If true, it uses a `foreach` loop to iterate over each user and write their details to the console. If false, it writes a message indicating no users are in the system.

```
// Wyświetlanie listy użytkowników
1 odwołanie
public void ShowUsers()
{
    if (users != null && users.Count > 0)
    {
        foreach (var user in users)
        {
            Console.WriteLine($"ID: {user.ID}, Imię: {user.Name}, Nazwisko: {user.Surname}," +
                              $"Telefon: {user.PhoneNumber}");
        }
    }
    else
    {
        Console.WriteLine("Brak użytkowników w systemie.");
    }
}
```

Rysunek 4.16: Kod źródłowy listy użytkowników biblioteki

## 4.9 Wypożyczanie książek

Użytkownik może wypożyczyć książkę, jeśli jest dostępna.



```
D:\Aplikacja\ConsoleApp1\bin\Debug\net6.0\ConsoleApp1.exe
=== System Biblioteczny ===
1. Dodaj książkę
2. Edytuj książkę
3. Usuń książkę
4. Zarejestruj użytkownika
5. Wypożycz książkę
6. Zwróć książkę
7. Wyświetl listę książek
8. Wyświetl listę użytkowników
9. Wyjście
Wybierz opcję: 5
ID: 0, Tytuł: Windows Server 2022, Autor: Adam Nogły, Rok: 2023, Dostępność: True
Podaj ID książki do wypożyczenia: 0
Podaj ID użytkownika: 0
```

Rysunek 4.17: Wypożyczanie książki

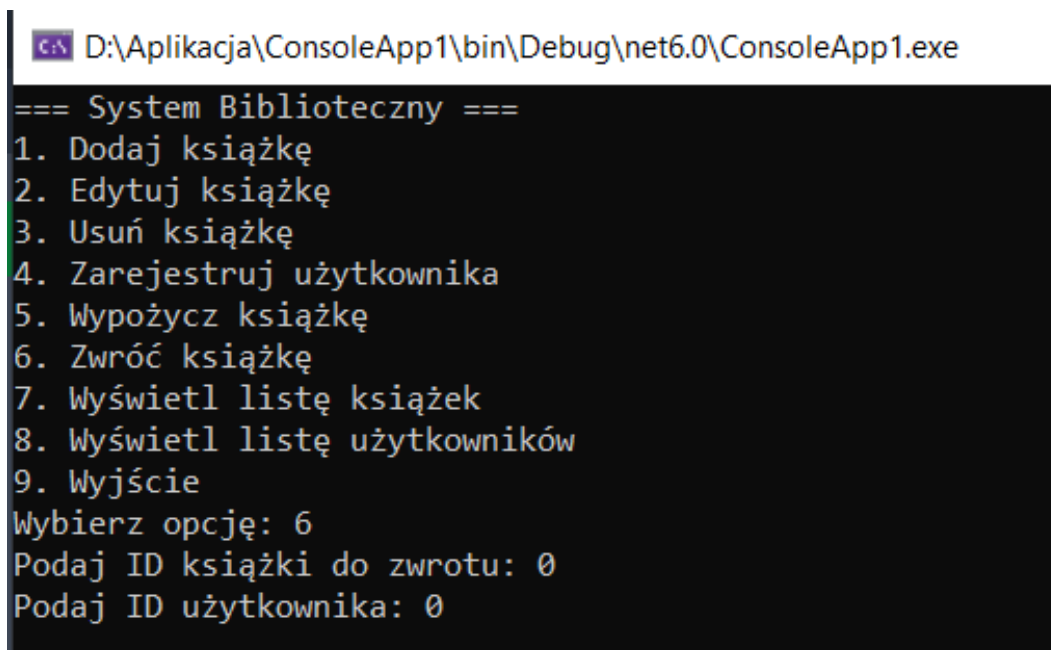
```
// Wypożyczanie książki dla użytkownika
1 odwołanie
public void BorrowBook(int bookId, int userId)
{
    Book? book = books.FirstOrDefault(b => b.ID == bookId);
    if (book != null && book.IsAvailable)
    {
        book.IsAvailable = false;
        transactions.Add(new Transaction
        {
            BookID = bookId,
            UserID = userId,
            BorrowDate = DateTime.Now,
            ReturnDate = null
        });

        fileHandler.SaveBooksToFile(books, "books.csv");
        fileHandler.SaveTransactionsToFile(transactions, "transactions.csv");
    }
    else
    {
        Console.WriteLine("Książka jest niedostępna.");
    }
}
```

Rysunek 4.18: Kod źródłowy wypożyczania książki

## 4.10 Zwrot książki

Użytkownik może zwrócić wcześniej wypożyczoną książkę.



```
D:\Aplikacja\ConsoleApp1\bin\Debug\net6.0\ConsoleApp1.exe

=== System Biblioteczny ===
1. Dodaj książkę
2. Edytuj książkę
3. Usuń książkę
4. Zarejestruj użytkownika
5. Wypożycz książkę
6. Zwróć książkę
7. Wyświetl listę książek
8. Wyświetl listę użytkowników
9. Wyjście
Wybierz opcję: 6
Podaj ID książki do zwrotu: 0
Podaj ID użytkownika: 0
```

Rysunek 4.19: Zwrot książki

```
// Zwrot książki do biblioteki
1 odwołanie
1 public void ReturnBook(int bookId, int userId)
1 {
1     Transaction? transaction = transactions.FirstOrDefault(t => t.BookID == bookId &&
1     t.UserID == userId && t.ReturnDate == null);
1     if (transaction != null)
1     {
1         transaction.ReturnDate = DateTime.Now;
1         Book? book = books.FirstOrDefault(b => b.ID == bookId);
1         if (book != null)
1         {
1             book.IsAvailable = true;
1         }
1         fileHandler.SaveBooksToFile(books, "books.csv");
1         fileHandler.SaveTransactionsToFile(transactions, "transactions.csv");
1     }
1     else
1     {
1         Console.WriteLine("Nie znaleziono transakcji wypożyczenia tej książki.");
1     }
1 }
```

Rysunek 4.20: Kod źródłowy zwrotu książki



## **4.11 Podsumowanie**

Rozdział ten prezentuje warstwę użytkową systemu zarządzania biblioteką. Opisano w nim podstawowe funkcjonalności oraz interfejs użytkownika, ilustrowany zrzutami ekranu przedstawiającymi pracę aplikacji. Każda operacja została szczegółowo omówiona, co umożliwia łatwiejsze zrozumienie obsługi systemu.

# Bibliografia

- [1] Grzybowski J., *C# 10 i .NET 6. Tworzenie nowoczesnych aplikacji dla platformy Windows*, Helion, 2022.
- [2] Zelenski T., *Bazy danych. Podstawy projektowania i implementacji*, PWN, 2021.
- [3] Ciszewski A., *Systemy informatyczne w bibliotece*, SBP, 2019.
- [4] Nowak P., Kowalska A., *Biblioteki cyfrowe i systemy zarządzania zasobami bibliotecznymi*, Przegląd Bibliotekarski, vol. 88, no. 2, 2023, s. 75-91.
- [5] Wydawnictwo Helion, *C# 10 i .NET 6 - Dokumentacja i kursy*, 2023. Dostępne online: <https://helion.pl/kategorie/csharp>
- [6] Wydawnictwo PWN, *Bazy danych - poradniki i kursy online*, 2023. Dostępne online: <https://pwn.pl/bazy-danych>
- [7] Stowarzyszenie Bibliotekarzy Polskich, *Systemy biblioteczne w Polsce - aktualne rozwiązania*, 2023. Dostępne online: <https://sbp.pl/systemy-biblioteczne>

# Spis rysunków

2.1	Diagram klas – hierarchia dziedziczenia . . . . .	9
2.2	Diagram klas – klasy wspierające i ich metody . . . . .	9
3.1	Diagram Gantta – harmonogram realizacji projektu . . . . .	12
4.1	Strona startowa aplikacji . . . . .	14
4.2	Kod źródłowy obsługi menu głównego . . . . .	14
4.3	Kod źródłowy obsługi menu głównego . . . . .	15
4.4	Kod źródłowy obsługi menu głównego . . . . .	16
4.5	Dodawanie nowej książki . . . . .	17
4.6	Kod źródłowy dodawania książki . . . . .	17
4.7	Edycja informacji o książce . . . . .	18
4.8	Kod źródłowy edycji książki . . . . .	18
4.9	Usuwanie książki . . . . .	19
4.10	Kod źródłowy usuwania książki . . . . .	19
4.11	Lista książek w bibliotece . . . . .	20
4.12	Kod źródłowy wyświetlania listy książek . . . . .	20
4.13	Dodawanie nowego użytkownika . . . . .	21
4.14	Kod źródłowy dodawania użytkownika . . . . .	21
4.15	Lista użytkowników biblioteki . . . . .	22
4.16	Kod źródłowy listy użytkowników biblioteki . . . . .	22
4.17	Wypożyczanie książki . . . . .	23
4.18	Kod źródłowy wypożyczania książki . . . . .	23
4.19	Zwrot książki . . . . .	24
4.20	Kod źródłowy zwrotu książki . . . . .	24

# Spis tabel

3.1	Harmonogram realizacji projektu – podział na etapy . . . . .	11
-----	--	----