

# PROGRAMOWANIE OBIEKTOWE

INEW00003P

## Projekt

Wydział Elektroniki	Kierunek: Informatyka
Grupa zajęciowa (np. Pn7:30): Środa 18:55	Semestr: 2017/18 LATO
Nazwisko i Imię: Maksim Birszel	Nr indeksu: 241353
Nr. grupy projektowej: INEW00003P	
Prowadzący: mgr inż. Piotr Lechowicz	

TEMAT:

*System biblioteki*

OCENA:

PUNKTY:

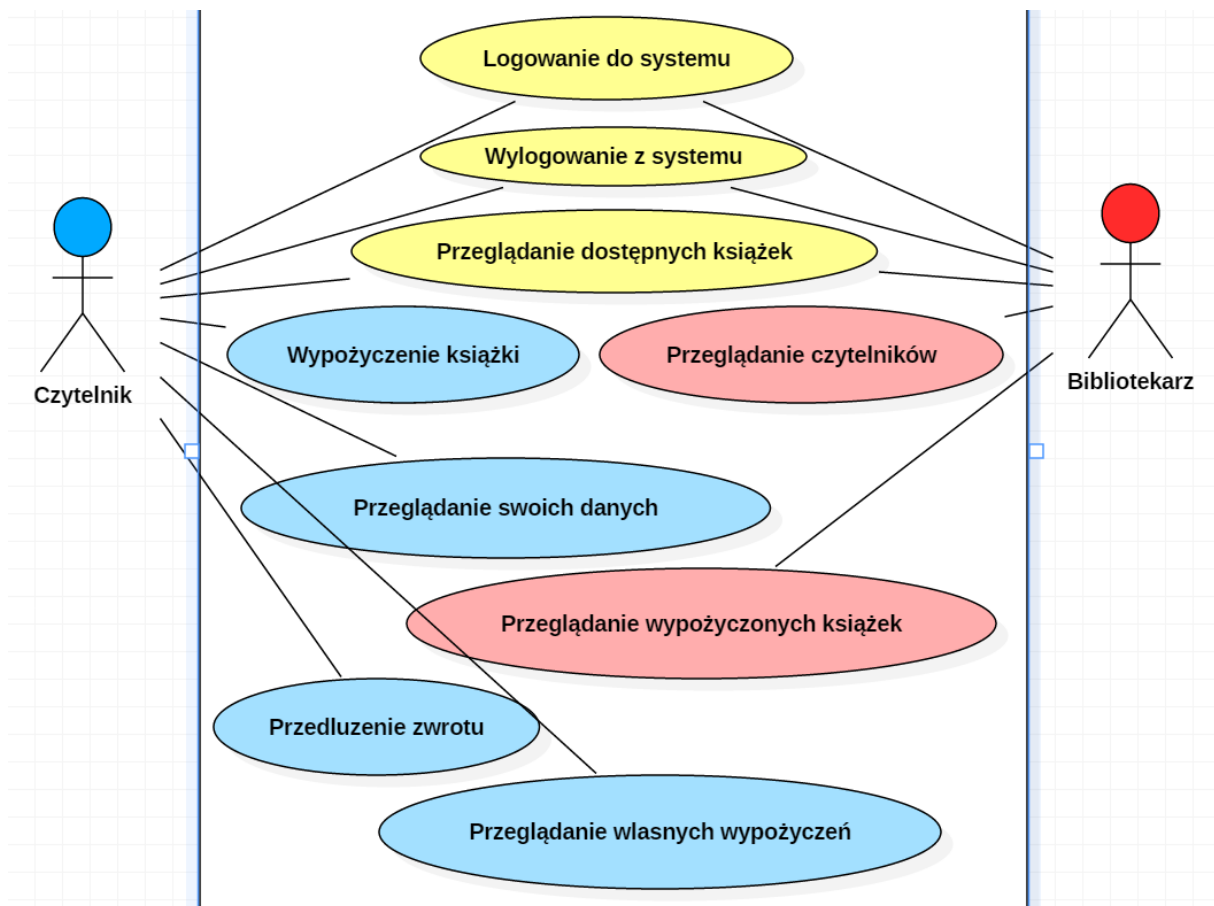
Data:

## 1. Założenia i opis funkcjonalny programu

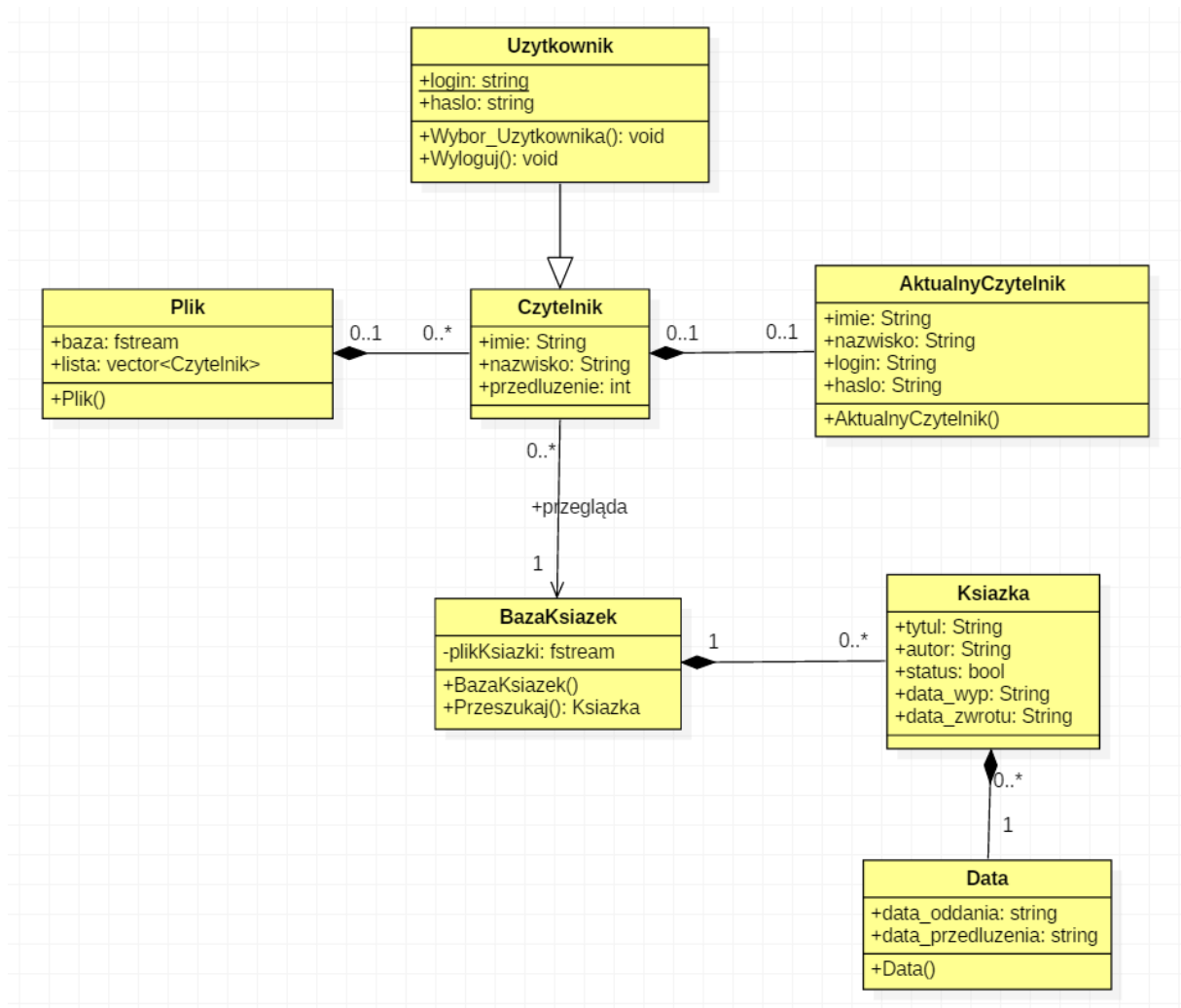
Program „System biblioteki” jest uproszczoną wersją typowego internetowego systemu do zarządzania oraz korzystania z biblioteki. Można z niego korzystać dwójako, jako czytelnik, bądź bibliotekarz. Posiada on bazę składającą się z kilkudziesięciu książek, którymi możemy operować. Każdy nowy użytkownik ma możliwość założenia konta w systemie. Po zalogowaniu się możemy przeglądać dostępne książki, wypożyczać je, oraz przedłużać zwrot. Możemy również zobaczyć nasze podstawowe dane logowania (imię, nazwisko, login, hasło) oraz krótką informację o programie. Książek nie można zwracać.

## 2. Diagramy UML

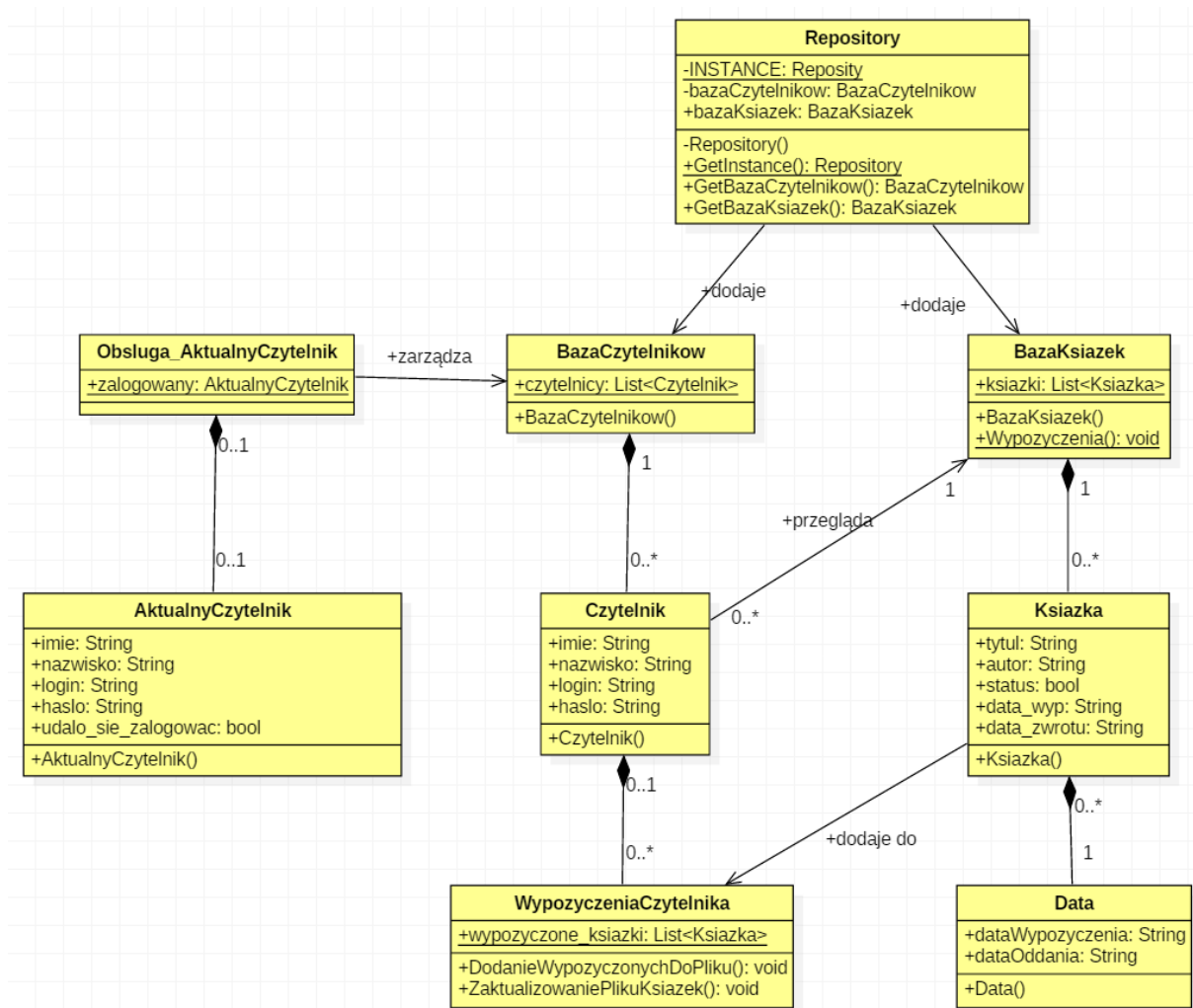
### a) Diagram przypadków użycia



## b) Diagram klas C++



### c) Diagram klas C#



## 3. Kody klas

### a) C++

```

// klasa tworząca obiekt będący obecnie zalogowanym użytkownikiem
// obiekt tworzy się w momencie zalogowania do systemu
class AktualnyCzytelnik
{
public:
    // dane obecnie zalogowanego użytkownika
    string imie;
    string nazwisko;
    string login;
    string haslo;
    AktualnyCzytelnik(string im, string naz, string log, string has);
};
    
```

```

// klasa odpowiedzialna za przechowywanie i obsługę wszystkich książek
class BazaKsiazek
{
// zmienna przechowująca plik z wszystkimi książkami
    fstream plikKsiazki;

public:
// konstruktor tworzący obiekt klasy przechowujący vector z wszystkimi książkami
// inicjalizuje się od razu po uruchomieniu programu
    BazaKsiazek(vector<Ksiazka> &ksiazki);
// funkcja przeszukująca vector z książkami i zwracająca jedną z wybranych pozycji
    Ksiazka przeszukaj(vector<Ksiazka> &ksiazki);
};

// klasa przechowująca imię, nazwisko, i ilość przedłużeń (wypożyczonych książek) każdego
// czytelnika, dziedziczy login i hasło z klasy Uzytkownik
class Czytelnik : public Uzytkownik
{
public:
// dane czytelnika
    string imie;
    string nazwisko;
// ilość przedłużeń książek wykonanych przez czytelnika
    int przedluzenie;
};

// klasa przechowująca własności książek
class Ksiazka
{
public:
// dane każdej książki
    string tytul;
    string autor;
// status wypożyczenia, true – wypożyczona, false - nie
    bool status;
// data wypożyczenia i zwrotu książki
    string data_wyp;
    string data_zwrotu;
};

// klasa odpowiedzialna za obsługę pliku przechowującego wszystkich czytelników
class Plik
{
public:
// zmienna przechowująca plik z czytelnikami
    fstream baza;
// vector tworzący liste czytelników
    vector<Czytelnik> lista;
// konstruktor tworzący obiekt klasy, który wczytuje czytelników z pliku do vectora
// tworzy się automatycznie po uruchomieniu programu
    plik(vector<Czytelnik> &lista);
};

```

*// klasa przechowująca login i hasło każdego zarejestrowanego czytelnika*

```
class Uzytkownik
{
public:
// login i hasło czytelnika
    string login;
    string haslo;
// funkcja wyszukiwająca czytelnika o podany loginie i hasle w bazie
    void Wybor_Uzytkownika();
// wylogowanie z systemu
    void Wyloguj();
};
```

## **b) C#**

*// klasa przechowująca dane aktualnie zalogowanego użytkownika*  
*// jej obiekt tworzy i inicjalizuje się automatycznie po zalogowaniu do systemu*

```
public class AktualnyCzytelnik
{
// dane obecnego czytelnika
    public string imie { get; set; }
    public string nazwisko { get; set; }
    public string login { get; set; }
    public string haslo { get; set; }
// zmienna która po udanym logowaniu zmienia wartość na true
    public bool udalo_sie_zalogowac = false;

    public AktualnyCzytelnik(string login, string haslo)
}
}
```

*// klasa przechowująca listę wszystkich czytelników*

```
public class BazaCzytelnikow
{
// statyczna lista czytelników
    public static List<Czytelnik> czytelnicy = new List<Czytelnik>();
// konstruktor wczytujący czytelników do listy
    public BazaCzytelnikow()
}
}
```

*// klasa przechowująca listę wszystkich książek*

```
public class BazaKsiazek
{
// statyczna lista książek
    public static List<Ksiazka> ksiazki = new List<Ksiazka>();
// konstruktor wczytujący książki do listy
    public BazaKsiazek()
// statyczna funkcja dodająca wypożyczone książki do listy wszystkich książek
    public static void Wypozyczenia()
}
}
```

*// klasa przechowująca dane każdego czytelnika pobierane z pliku*

```
public class Czytelnik
{
    public string imie { get; set; }
    public string nazwisko { get; set; }
    public string login { get; set; }
    public string haslo { get; set; }
}
```

```

// konstruktory wczytujące czytelników z pliku
    public Czytelnik() { }
    public Czytelnik(string imie, string nazwisko, string login, string haslo)
    }

// klasa zarządzająca datami wypożyczeń i zwrotów książek
public class Data
{
    // zmienna przechowująca dzisiejszą datę
    public string dataWypozyczenia;
    // zmienna przechowująca datę o miesiąc późniejszą od dnia dzisiejszego
    public string dataOddania;
    // konstruktor wczytujący daty do zmiennych dataWypozyczenia oraz dataOddania oraz
    // sprawdzający czy dataOddania po dodaniu miesiąca nie będzie nieprawidłowa
    // np. 01.12.18 + 1 miesiąc daje nam 01.13.18, a taka data nie istnieje
    public Data()
    }

// klasa przechowująca dane książek
public class Ksiazka
{
    public string tytul { get; set; }
    public string autor { get; set; }
    // gdy na stanie - true, po wypożyczeniu - false
    public bool status { get; set; }
    // data wypożyczenia, na początku ma wartość „na stanie”, po wypożyczeniu zmienia się
    // na datę dnia, w którym została wypożyczona
    public string data_wyp { get; set; }
    // data oddania, na początku ma wartość „brak”, po wypożyczeniu zmienia się na datę
    // miesiąc późniejszą niż data dnia, w którym została wypożyczona
    public string data_zwrotu { get; set; }
    // konstruktory przypisujące zmienne do książek
    public Ksiazka() { }
    public Ksiazka(string tytul, string autor)
    public Ksiazka(string tytul, string autor, string data_wyp, string data_zwrotu)
    }

// klasa zarządzająca klasą aktualnego czytelnika
public class Obsluga_AktualnyCzytelnik
{
    // zmienna statyczna tworząca obiekt klasy AktualnyCzytelnik ze wszystkimi danymi
    // obecnej osoby oraz widoczny w całym programie
    public static AktualnyCzytelnik zalogowany;
    }

// klasa tworząca oraz inicjalizująca bazę czytelników oraz bazę książek, które
// będą widoczne w całym programie przez co nie będą musiały być tworzone na nowo
// w każdej formie
public class Repository
{
    private static Repository INSTANCE;

    private BazaCzytelnikow bazaCzytelnikow = new BazaCzytelnikow();
    public BazaKsiazek bazaKsiazek = new BazaKsiazek();

    private Repository() { }

    public static Repository GetInstance()

```

```

public BazaCzytelnikow GetBazaCzytelnikow()

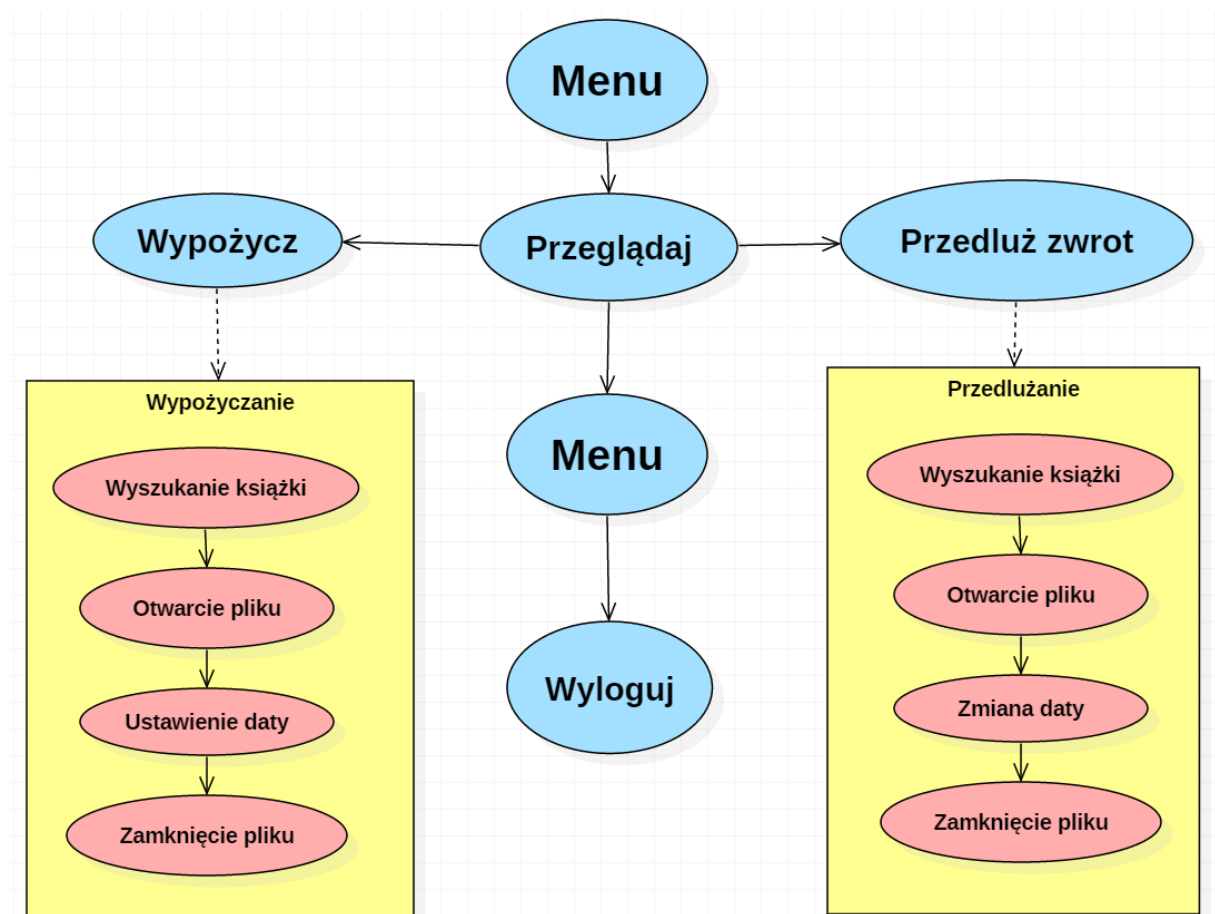
public BazaKsiazek GetBazaKsiazek()
}

// klasa przechowująca oraz zarządzająca wypożyczonymi książkami
public class WypozyczeniaCzytelnika
{
// zmienna statyczna tworząca listę z wypożyczonymi książkami
public static List<Ksiazka> wypozyzione_ksiazki = new List<Ksiazka>();

// funkcja dodająca wypożyczone książki do pliku wypożyczeń aktualnego czytelnika wraz
// z datami wypożyczenia i oddania
public static void DodanieWypozyczonychDoPliku()
// zaktualizowanie pliku z dostępnymi książkami poprzez zmianę daty wypożyczenia i
//oddania wybranej książki
public static void ZaktualizowaniePlikuKsiazek()
}

```

#### 4. Schematy blokowe oraz kod własnych funkcji





## a) C++

```
// konstruktor klasy BazaKsiazek wczytujący z pliku „plikKsiazki” wszystkie
// książki wraz z ich parametrami (autor, tytuł, data wypożyczenia, itp.)
BazaKsiazek::BazaKsiazek(vector <Ksiazka> &ksiazki)
{
    Ksiazka nowa;

    plikKsiazki.open("E:\\Projekt\\build-SystemBiblioteki-
Desktop_Qt_5_8_0_MinGW_32bit-Debug\\debug\\plikKsiazki.txt");

    if(plikKsiazki.good() == false)
    {
        QApplication->quit();
    }

    int elementy = ksiazki.size();
    //wczytywanie każdego stringa po kolei z pliku do zmiennych
    for(int i=0; i < elementy; i++)
    {
        plikKsiazki >> nowa.tytul >> nowa.autor >> nowa.status >>
nowa.data_wyp >> nowa.data_zwrotu;
        ksiazki.push_back(nowa);
    }

    plikKsiazki.close();
}

//funkcja przeszukująca bazę czytelników i porównywanie ich z wpisanymi
//danymi logowania przez aktualnego użytkownika
//jeśli dane się zgadzają z którymś z zapisanych czytelników to użytkownik
//zostaje zalogowany do systemu
void LogowanieCzytelnik1()
{
    BibliotekaCzytelnik *oknoCzytelnika;
    oknoCzytelnika = new BibliotekaCzytelnik();
    fstream osoba;
    string im, naz, log, has;

    osoba.open("E:\\Projekt\\build-SystemBiblioteki-
Desktop_Qt_5_8_0_MinGW_32bit-Debug\\debug\\baza.txt");

    if(osoba.good()==false)
    {
        QApplication->quit();
    }

    while(!osoba.eof())
    {
        osoba >> im >> naz >> log >> has;

        if(ui->czyt_login->text().toStdString() == log && ui->czyt_haslo-
>text().toStdString() == has)
        {
            AktualnyCzytelnik(im, naz, log, has);
        }
    }
}
```

```

// zapisanie danych zalogowanego uzytkownika w pliku
    hide();
    oknoCzytelnika->show();
// zalogowanie sie czytelnika posiadajacego konto do bazy
    osoba.close();
}

}

    if(ui->czyt_login->text().toStdString() != log && ui->czyt_haslo-
>text().toStdString() != has)
    {
        osoba.close();
        QMessageBox::information(this, "Logowanie", "Błędny login lub
hasło, sprawdź czy na"
                                "pewno posiadasz
konto");
        LogowanieCzytelnik logowCzyt;
        hide();
        logowCzyt.setModal(true); //ponowne otworenie konta po podanych
blednym loginie lub hasle
        logowCzyt.exec();
    }
}

//funkcja wywoływana podczas rejestracji uzytkownika nie posiadajacego
//konta w systemie
//zapisuje ona uzytkownika o podanych loginie, hasle, imieniu i nazwisko
//do bazy, a nastepnie przekierowuje do okna logowania
void LogowanieCzytelnik2()
{
    Czytelnik kolejny;
    fstream baza;
    vector <Czytelnik> lista;
    baza.open("E:\\Projekt\\build-SystemBiblioteki-
Desktop_Qt_5_8_0_MinGW_32bit-Debug\\debug\\baza.txt", ios::app);

    if(baza.good() == false)
    {
        qApp->quit();
    }

    kolejny.imie = ui->podajImie->text().toStdString();
    kolejny.nazwisko = ui->podajNazwisko->text().toStdString();
    kolejny.login = ui->podajLogin->text().toStdString();
    kolejny.haslo = ui->podajHaslo->text().toStdString();
//dodawanie danych podanych przez czytelnika do pliku
    baza << endl << kolejny.imie << " " << kolejny.nazwisko << " " <<
kolejny.login << " " << kolejny.haslo;

    lista.push_back(kolejny);

    baza.close();

    QMessageBox::information(this, "Poprawna rejestracja", "Rejestracja
przebiegła pomyślnie, zaloguj się do systemu");

    LogowanieCzytelnik logowCzyt;
    hide();
    logowCzyt.setModal(true); //ponowne otworenie konta po poprawnej
rejestracji

```

```

        logowCzyt.exec();
    }

//funkcja dodajaca czytelnikow z pliku do wektora
//uruchamia się na starcie programu, aby można było operować na
//czytelnikach bez ciągłego wczytywania ich z pliku
plik::plik(vector<Czytelnik> &lista)
{
    Czytelnik nowy;

    baza.open("E:\\Projekt\\SystemBiblioteki\\baza.txt");

    if(baza.good() == false)
    {
        qApp->quit();
    }

    int ile = lista.size();

    for(int i=0; i<ile; i++)
    {
        baza >> nowy.imie >> nowy.nazwisko >> nowy.login >> nowy.haslo;
        lista.push_back(nowy);
    }

    baza.close();
}

//funkcja klasy PrzegladaJksiazki zwracajaca zmienną typu string,
//która jest datą wypożyczenia książki (aktualną datą)
string PrzegladaJksiazki::date()
{
    SYSTEMTIME stime;
    GetLocalTime(&stime);
    char buf[40] = {0};
    //przekonwertowanie daty na stringa
    sprintf(buf,"%02d/%02d/%04d", stime.wDay, stime.wMonth, stime.wYear);
    string dt = buf;
    return dt;
}

//funkcja podobna do powyższej, ta z kolei ma za zadanie ustawianie daty
//oddania książki, czyli daty miesiąc późniejszej od poprzedniej
string PrzegladaJksiazki::data_oddania()
{
    SYSTEMTIME stime;
    GetLocalTime(&stime);

    char buff[40] = {0};

    sprintf(buff,"%02d/%02d/%04d", stime.wDay, stime.wMonth+1,
stime.wYear);

    string oddanie = buff;

    return oddanie;
}

//funkcja przedłużająca datę oddania książki o 5 dni i 3 miesiące
string PrzegladaJksiazki::data_przedluzenia()

```

```

{
    SYSTEMTIME stime;
    GetLocalTime(&stime);

    char buff[40] = {0};

    sprintf(buff, "%02d/%02d/%04d", stime.wDay+5, stime.wMonth+3,
stime.wYear);

    string oddanie = buff;

    return oddanie;
}

```

## b) C#

```

// jest to konstruktor przyjmujący za parametry login i hasło zalogowanego czytelnika
// jego zadaniem jest otworenie pliku z zarejestrowanymi czytelnikami, a następnie
// przeszukanie go, po znalezieniu czytelnika o podanym loginie i hasle, wczytuje
// jego imie i nazwisko
// konstruktor jest wyposażony w blok wyłapywania wyjątków try - catch
// dzięki temu nie trzeba sprawdzać czy plik został otwarty poprawnie oraz czy nie
// jest on pusty, itp.
public AktualnyCzytelnik(string login, string haslo)
{
    // otworzenie pliku z czytelnikami w formie „do odczytu”
    FileStream plikCzytelnicy = new FileStream("E:\\Projekt
C#\\SystemBiblioteki\\SystemBiblioteki\\pliki\\plikCzytelnicy.txt", FileMode.Open,
FileAccess.Read);
    StreamReader odczyt = new StreamReader(plikCzytelnicy);

    try
    {
        // sprawdzenie czy użytkownik o podanych danych logowania istnieje w bazie czytelników
        // jeśli tak to przypisuje jego imię, nazwisko, login i hasło obiektowi klasy
        // i otwiera system biblioteki dla czytelnika o podanych danych logowania
        while (!odczyt.EndOfStream)
        {
            string wiersz_imie = odczyt.ReadLine();
            string wiersz_nazwisko = odczyt.ReadLine();
            string wiersz_login = odczyt.ReadLine();
            string wiersz_haslo = odczyt.ReadLine();

            // porównanie wpisanego loginu i hasła z danymi w pliku
            // jeśli są poprawne to zmienne klasy Czytelnik zostają nimi zainicjalizowane
            if (login.Equals(wiersz_login) && haslo.Equals(wiersz_haslo))
            {
                this.imie = wiersz_imie;
                this.nazwisko = wiersz_nazwisko;
                this.login = wiersz_login;
                this.haslo = wiersz_haslo;
                this.udalo_sie_zalogowac = true;

                break;
            }
        }

        if(!udalo_sie_zalogowac)
        {

```

```

        MessageBox.Show("Błędny login lub hasło! Spróbuj ponownie.");
    }
}
catch(Exception wyjatek)
{
    MessageBox.Show(wyjatek.ToString());
    Application.Exit();
}
finally
{
    odczyt.Close();
    plikCzytelnicy.Close();
}
}
// podobnie zbudowane są konstruktory BazaKsiazek oraz BazaCzytelników, dlatego
// nie będę ich drugi raz wklejał, bo różnią się głównie danymi

// funkcja Wypozyczenia należy do klasy BazaKsiazek, nie zwraca ona żadnych wartości
// oraz nie przyjmuje parametrów
// jej zadaniem jest dodawanie na bieżąco wypożyczonych przez użytkownika książek
// do listy wypożyczone_ksiazki, z której są później dodawane do osobistego pliku
// z wypożyczeniami każdego czytelnika
public static void Wypozyczenia()
{
    // stworzenie zmiennej dane_czytelnika, która od teraz będzie nazwą pliku z
    // wypożyczeniami każdego czytelnika, np. Piotr_Kowalski.txt
    // imię oraz nazwisko są wczytywane z obiektu AktualnyCzytelnik
    string imie = Obsluga_AktualnyCzytelnik.zalogowany.imie;
    string nazwisko = Obsluga_AktualnyCzytelnik.zalogowany.nazwisko;
    string dane_czytelnika = imie + "_" + nazwisko;
    // stworzenie osobistego pliku z wypożyczeniami czytelnika, bądź jeśli taki istnieje
    // to otwarcie go w formie do „zapisu i odczytu”
    FileStream plik_wypozycone = new FileStream("E:\\Projekt
C#\\SystemBiblioteki\\SystemBiblioteki\\pliki\\wypozyczenia\\" + dane_czytelnika +
".txt", FileMode.OpenOrCreate, FileAccess.ReadWrite);
    StreamReader odczyt = new StreamReader(plik_wypozycone);
    // stworzenie obiektu klasy WypozyczeniaCzytelnika
    WypozyczeniaCzytelnika wypozyczenia = new WypozyczeniaCzytelnika();

    try
    {
        // dopisywanie wszystkich książek z pliku z wypożyczeniami do
        // listy - wypożyczone_ksiazki
        while (!odczyt.EndOfStream)
        {
            string tytul, autor, data_wyp, data_zwr;

            tytul = odczyt.ReadLine();
            autor = odczyt.ReadLine();
            data_wyp = odczyt.ReadLine();
            data_zwr = odczyt.ReadLine();

            Ksiazka ksiazka = new Ksiazka(tytul, autor, data_wyp, data_zwr);
            WypozyczeniaCzytelnika.wypozycone_ksiazki.Add(ksiazka);
        }
    }
    catch (Exception wyjatek)
    {
        MessageBox.Show(wyjatek.ToString());
    }
}

```

```

        Application.Exit();
    }
    finally
    {
        odczyt.Close();
    }
}

// kolejny konstruktor, ma on za zadanie tworzyć obiekt klasy Data i przypisywać
// zmiennej dataWypozyczenia - obecną datę
// a zmiennej dataOddania - obecną datę powiększoną o 1 miesiąc
// konstruktor ponadto jest wyposażony w mechanizm sprawdzania czy powiększona
// o miesiąc data nie jest nieprawidłowa, np. gdy mamy 01.12.18, to po powiększeniu
// dostaniemy 01.13.18, w takim wypadku miesiąc zostaje zmieniony na 01, a rok
// na o jeden większy
public Data()
{
    // wczytanie obecnej daty do zmiennej data
    DateTime data = DateTime.Now;
    // przypisanie obecnej daty zmiennej dataWypozyczenia w formie dd.mm.rr
    this.dataWypozyczenia = (data.Day.ToString() + "." + data.Month.ToString()
+ "." + data.Year.ToString());
    // sprawdzenie poprawności daty oraz jej poprawienie jeśli to konieczne
    if(data.Month.ToString().Equals("12"))
    {
        this.dataOddania = (data.Day.ToString() + ".1." + (data.Year + 1));
    }
    else
    {
        this.dataOddania = (data.Day.ToString() + "." + (data.Month + 1) + "."
+ data.Year.ToString());
    }
}

// funkcja dodająca wypożyczone książki z listy do osobistego pliku czytelnika
// jest to funkcja statyczna, nie przyjmuje parametrów i zwraca wartość void
public static void DodanieWypozyczonychDoPliku()
{
    string imie = Obsluga_AktualnyCzytelnik.zalogowany.imie;
    string nazwisko = Obsluga_AktualnyCzytelnik.zalogowany.nazwisko;
    string dane_czytelnika = imie + "_" + nazwisko;

    FileStream plik_wypozyczone = new FileStream("E:\\Projekt
C#\\SystemBiblioteki\\SystemBiblioteki\\pliki\\wypozyczenia\\" + dane_czytelnika +
".txt", FileMode.Truncate, FileAccess.Write);

    StreamWriter zapis = new StreamWriter(plik_wypozyczone);

    try
    {
        // blok foreach przeszukuje liste wypozyczone_ksiazki, a następnie zapisuje linia
        // po linii każdy parametr każdej książki
        foreach(Ksiazka ks in WypozyczeniaCzytelnika.wypozyczone_ksiazki)
        {
            zapis.WriteLine(ks.tytul.ToString());
            zapis.WriteLine(ks.autor.ToString());
            zapis.WriteLine(ks.data_wyp.ToString());
            zapis.WriteLine(ks.data_zwrotu.ToString());
        }
    }
}

```

```

        catch(Exception wyjatek)
        {
            MessageBox.Show(wyjatek.ToString());
        }
        finally
        {
            zapis.Close();
        }
    }

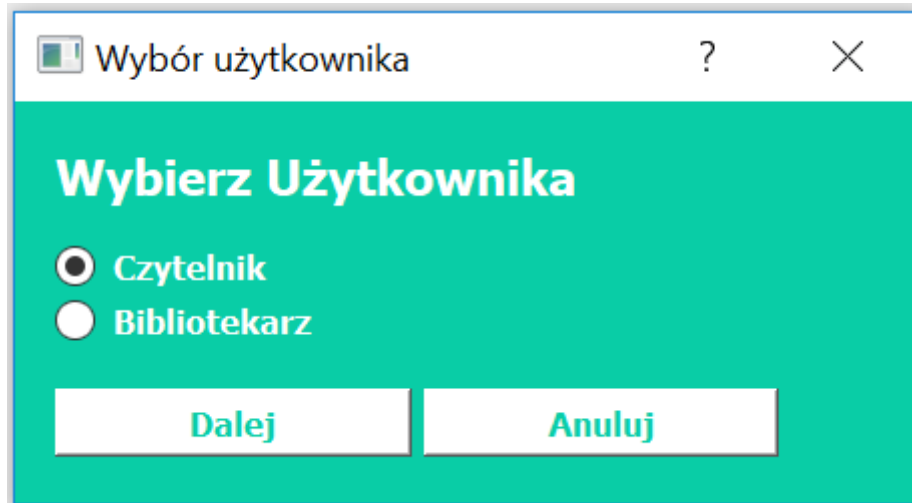
    // funkcja ZaktualizowaniePlikuKsiazek jest funkcją statyczną, nie przyjmuje
    // parametrów i zwraca wartość void, ma za zadanie aktualizację pliku z książkami
    // poprzez zmianę ich dat wypożyczeń oraz zwrotów z np. Symfonia C++, J. Grębosz,
    // na stanie, brak, na np. Symfonia C++, J. Grębosz, 13.03.18, 13.04.18
    public static void ZaktualizowaniePlikuKsiazek()
    {
        FileStream plikKsiazki = new FileStream("E:\\Projekt
C#\\SystemBiblioteki\\SystemBiblioteki\\pliki\\plikKsiazki.txt", FileMode.Truncate,
FileAccess.ReadWrite);
        StreamWriter zapis = new StreamWriter(plikKsiazki);

        try
        {
            foreach(Ksiazka ks in BazaKsiazek.ksiazki)
            {
                zapis.WriteLine(ks.tytul.ToString());
                zapis.WriteLine(ks.autor.ToString());
                zapis.WriteLine(ks.data_wyp.ToString());
                zapis.WriteLine(ks.data_zwrotu.ToString());
            }
        }
        catch (Exception wyjatek)
        {
            MessageBox.Show(wyjatek.ToString());
            Application.Exit();
        }
        finally
        {
            zapis.Close();
        }
    }
}

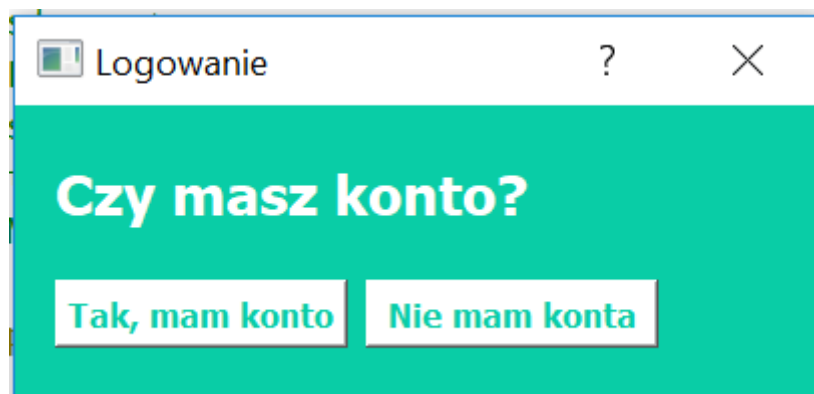
```

## 5. Opis użytkowy programu C++

Po uruchomieniu programu ukazuje nam się okno, w którym możemy wybrać jakim typem użytkownika jesteśmy. Po wybraniu „czytelnika” ukaże nam się okno pytające czy mamy już konto czy nie. Po wybraniu bibliotekarza automatycznie przejdziemy do okna logowania.

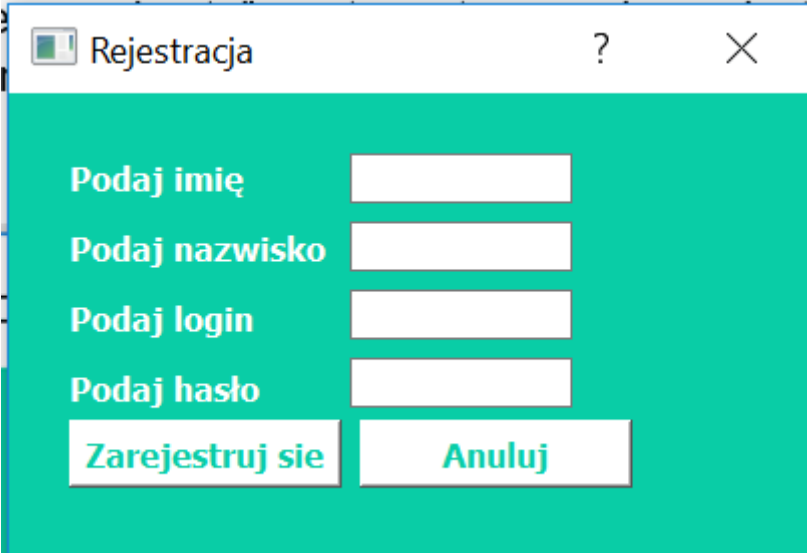


Gdy wybierzemy opcję „Nie mam konta”, system otworzy okno rejestracji, w którym będziemy mogli założyć konto. Po wybraniu „Tak, mam konto”, zostaniemy przeniesieni do okna logowania.



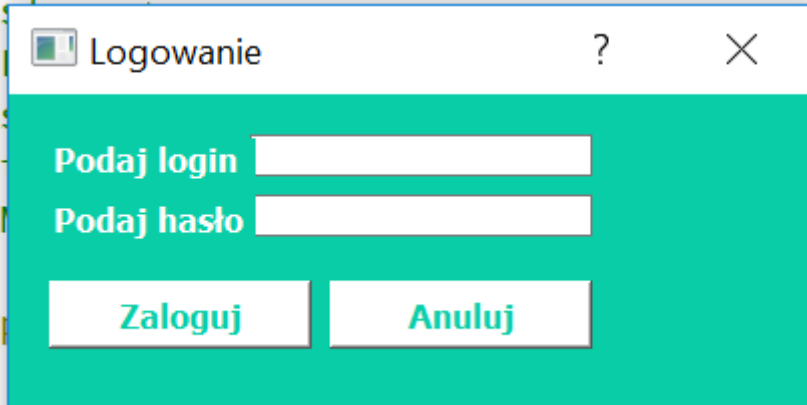


Aby założyć konto wpisujemy dane, a następnie klikamy przycisk zarejestruj się.



The image shows a registration window titled "Rejestracja". It has a light blue header bar with a question mark icon and a close button. The main area has a light blue background. There are four input fields with labels: "Podaj imię", "Podaj nazwisko", "Podaj login", and "Podaj hasło". Below the input fields are two buttons: "Zarejestruj się" and "Anuluj".

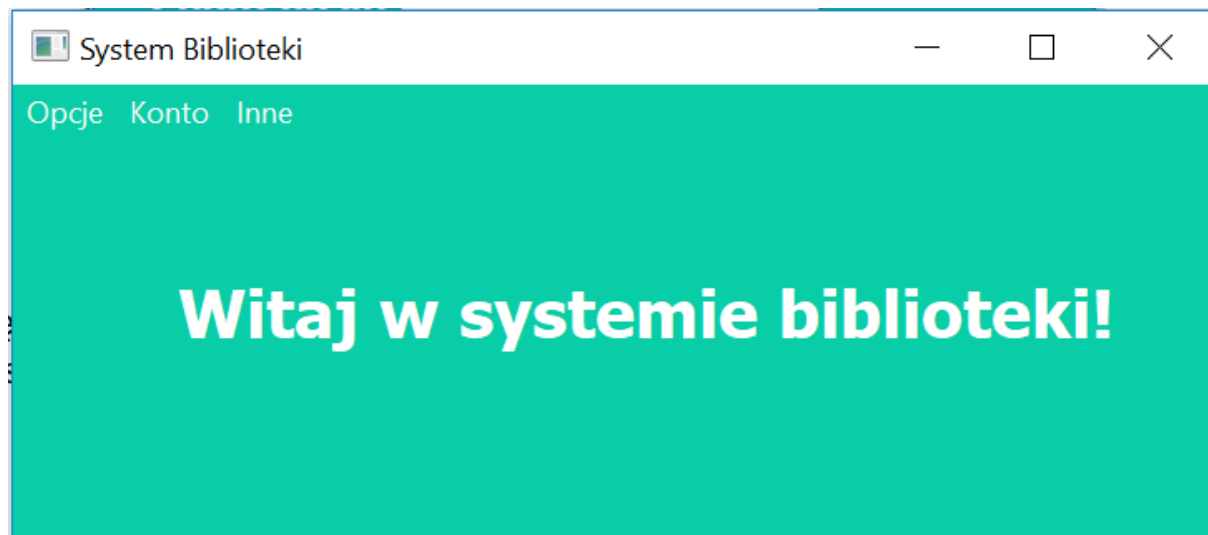
Po tak wykonanych czynnościach system przenosi nas do systemu logowania.



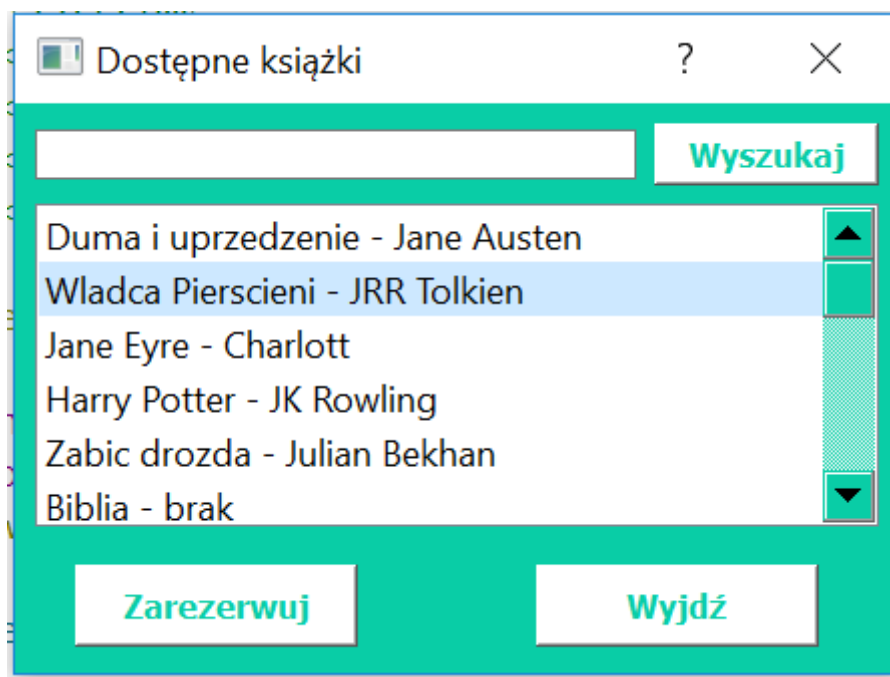
The image shows a login window titled "Logowanie". It has a light blue header bar with a question mark icon and a close button. The main area has a light blue background. There are two input fields with labels: "Podaj login" and "Podaj hasło". Below the input fields are two buttons: "Zaloguj" and "Anuluj".

Po wciśnięciu zaloguj, oraz podaniu poprawnych danych logowania, pokaże nam się menu główne systemu biblioteki.

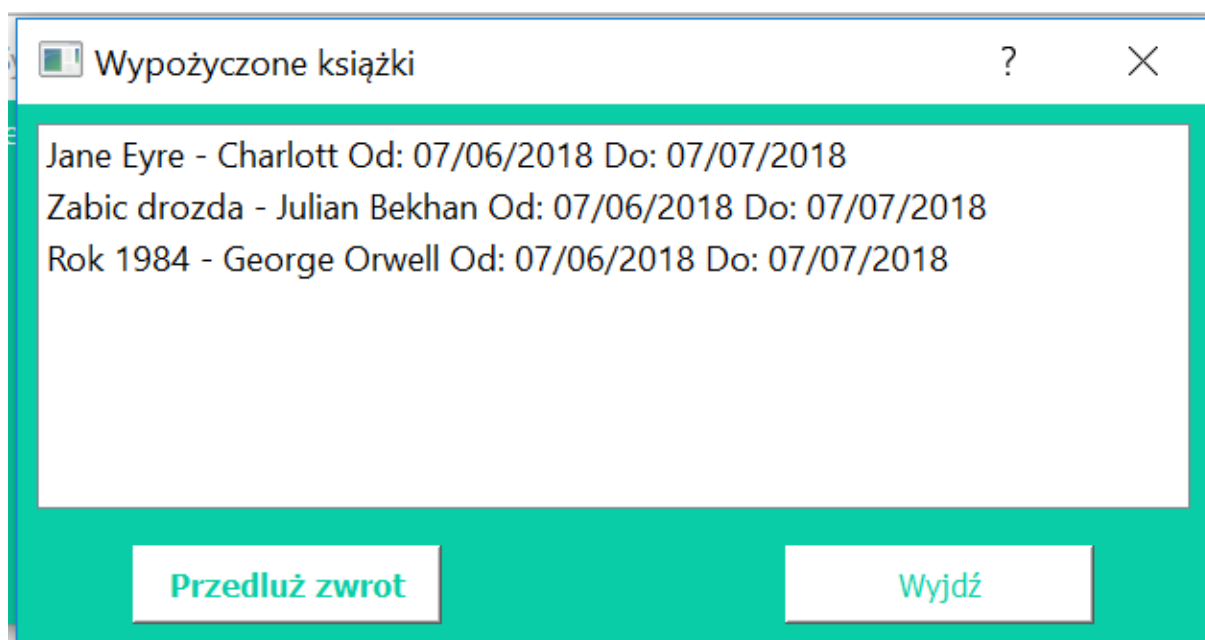
W głównym pasku narzędzi mamy do wyboru 3 okna: „Opcje”, „Konto” oraz „Inne”. Po wciśnięciu przycisku „Opcje” mamy do wyboru 3 warianty: „Przeglądaj książki”, „Wypożyczone Książki” i „Wyloguj”. W okienku „Konto” mamy opcje „Dane czytelnika”, a w zakładce „Inne” – „O programie”.



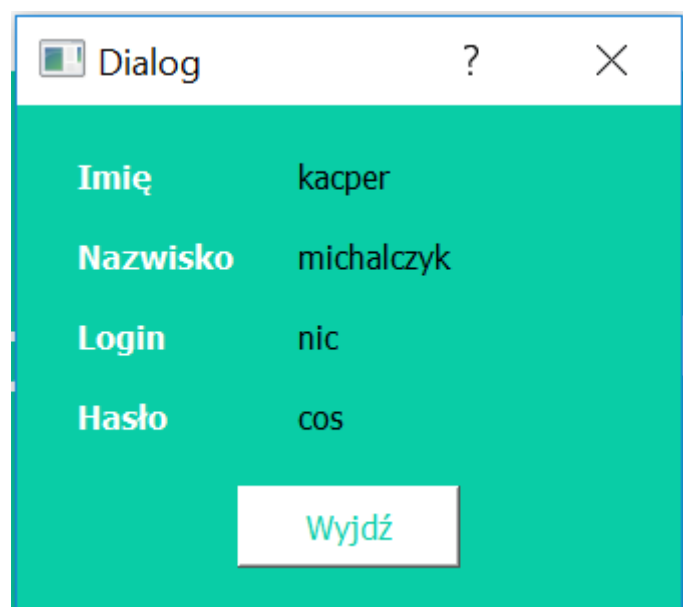
Po wybraniu „Przeglądaj książki” ukaże nam się okienko z wszystkimi książkami dostępnymi w bibliotece. Gdy chcemy wypożyczyć którąś książkę, należy wybrać książkę, a następnie kliknąć przycisk „Zarezerwuj”.



Po wybraniu „Wypożyczone książki” ukażą nam się książki, które wcześniej wypożyczyliśmy. Mamy teraz możliwość przedłużenia ich zwrotu, wybierając książkę, a następnie klikając przycisk „Przedłuż zwrot”.



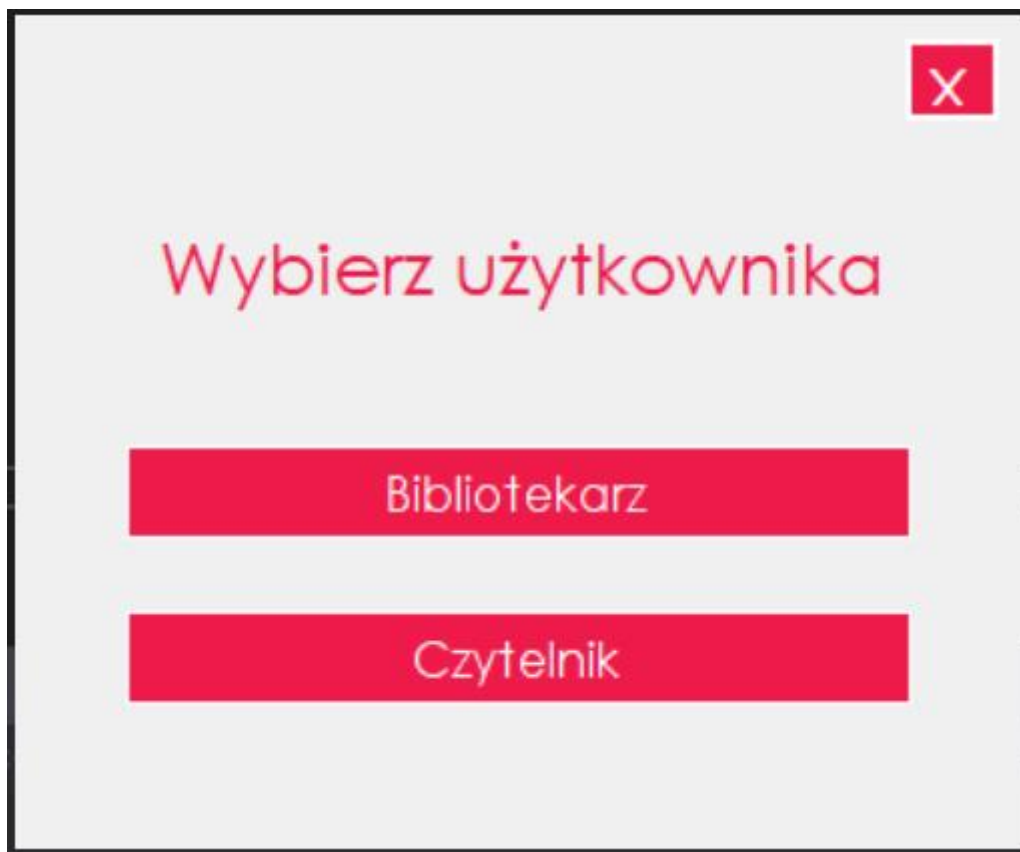
W „Dane czytelnika” możemy sprawdzić nasze dane logowania, które podaliśmy podczas rejestracji do systemu.



W okienku „O programie” wyświetla nam się krótka informacja o programie oraz autorze, które nie są istotne dla działania programu, więc zrezygnowałem z umieszczania ich z dokumentacji.


## Opis użytkowy programu C#


Po uruchomieniu programu pojawia nam się okienko wyboru użytkownika. Należy w nim wybrać jakim typem jesteśmy.



Po wybraniu jednej z dwóch opcji ukaże nam się okno logowania. Aby zalogować się do systemu wymagane jest podanie prawidłowych loginu i hasła, lub jeśli nie mamy konta to założenie go kliknięciem w przycisk „załóż konto” w prawym dolnym rogu okna.







System Biblioteki  
wersja v1

Zaloguj się

Login:


Hasło:

Zaloguj się

Jeśli nie masz jeszcze konta zarejestruj się!

Załącz konto

Aby się zarejestrować musimy podać poprawne dane i kliknąć przycisk „Zarejestruj się”.



System Biblioteki  
wersja v1

X

## Zarejestruj się

Imię:

Nazwisko:

Login:

Hasło:

☒ Akceptuje regulamin

Zarejestruj się

Po zalogowaniu ukaże nam się menu czytelnika.



# System biblioteki

X

Przeglądaj książki

Wypożyczone książki

Twoje dane

Informacje

Wyloguj

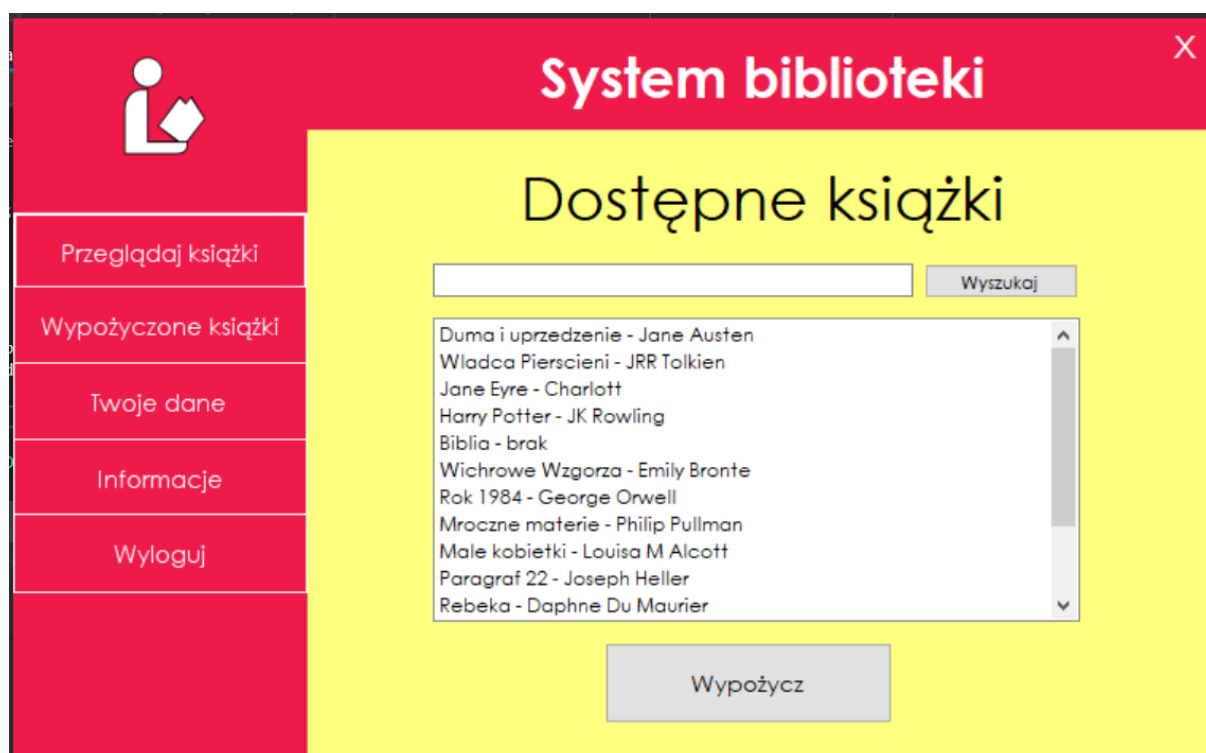
## SYSTEM BIBLIOTEKI

**System biblioteki wersja v1**

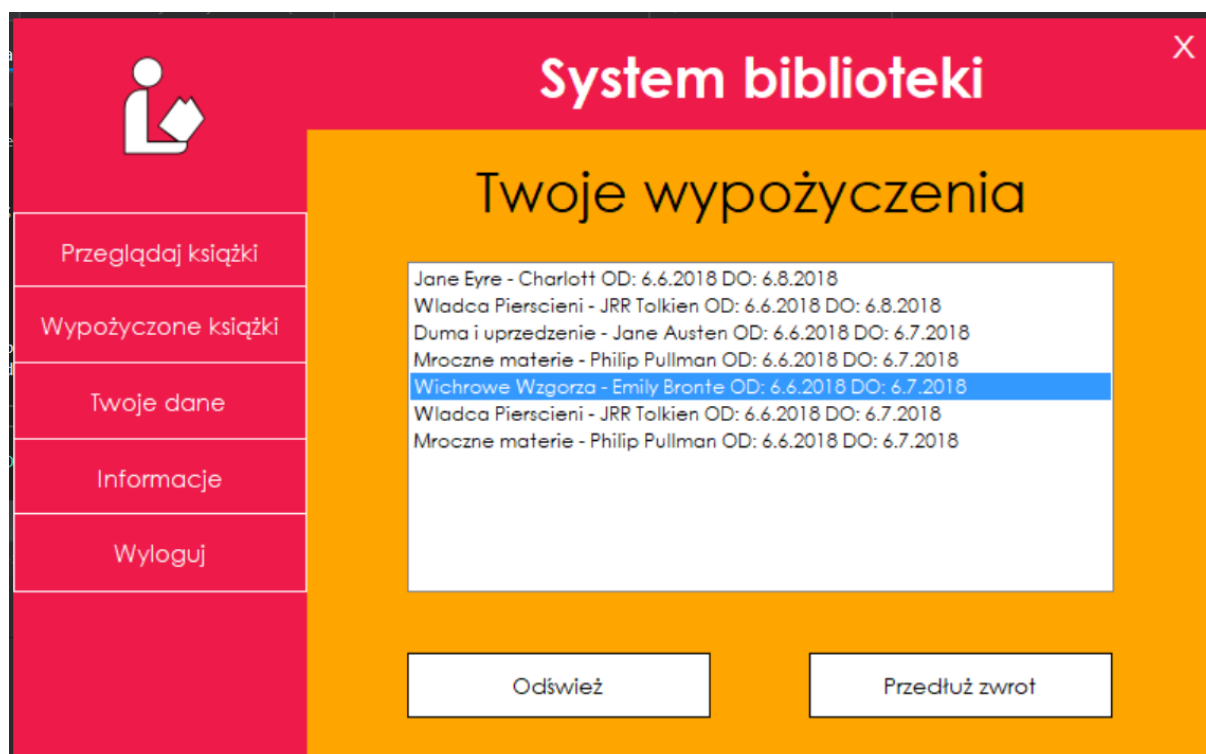
Autor aplikacji: Maksim Birszel



W zakładce „Przeglądaj książki” znajdują się wszystkie dostępne do wypożyczenia książki.



Gdy dokonamy wypożyczenia, po kliknięciu w „Wypożyczone książki” będziemy mogli zobaczyć książki, które zarezerwowaliśmy oraz przedłużyć ich zwrot. Po każdej operacji musimy odświeżyć okno aby książki się zaktualizowały.



W zakładce „Twoje dane” znajdują się dane, które wpisaliśmy podczas rejestracji.

The screenshot shows the 'System biblioteki' interface. On the left is a red sidebar with a white icon of a person reading a book. Below the icon are five menu items: 'Przeglądaj książki', 'Wypożyczone książki', 'Twoje dane' (highlighted), 'Informacje', and 'Wyloguj'. The main area has a red header with the title 'System biblioteki' and a close button 'X'. Below the header, the title 'Dane czytelnika' is centered. The user's details are displayed in two rows: 'Imię: kacper' and 'Nazwisko: michalczyk' in the first row, and 'Login: nic' and 'Hasło: nic' in the second row.

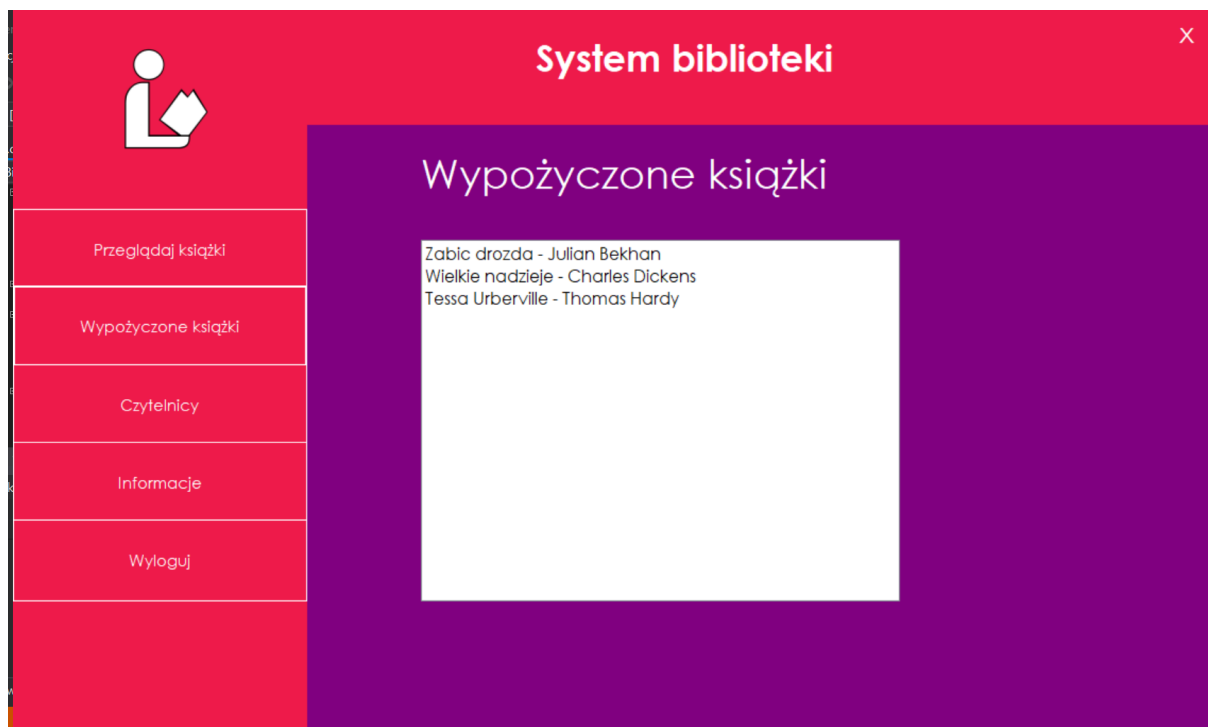
Aby zalogować się jako bibliotekarz wpisujemy login „admin” oraz hasło „admin”. Wtedy ukaże nam się menu bibliotekarza. W zakładce „Przeglądaj książki” możemy obejrzeć wszystkie niewypożyczone pozycje dostępne w bibliotece.

The screenshot shows the 'System biblioteki' interface with the 'Przeglądaj książki' menu item highlighted in the red sidebar. The main area has a red header with the title 'System biblioteki' and a close button 'X'. Below the header, the title 'Książki' is centered. A search bar with a 'Wyszukaj' button is located below the title. A list of books is displayed in a scrollable box:

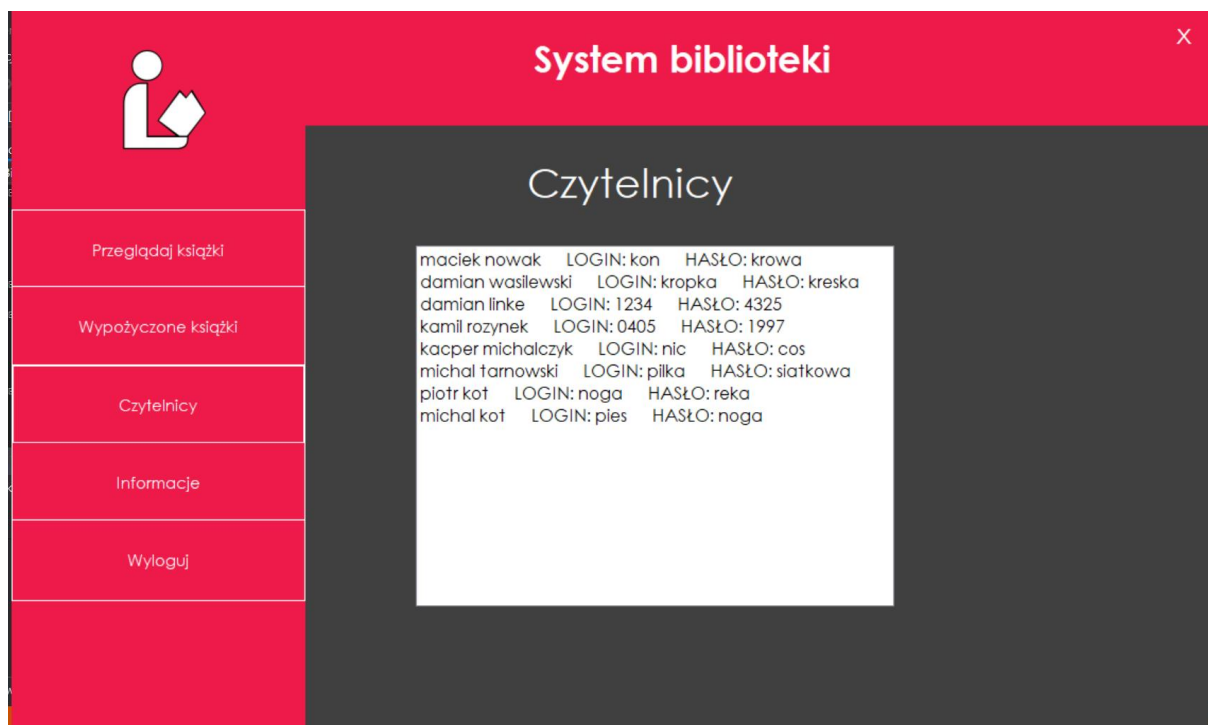
- Duma i uprzedzenie - Jane Austen
- Władca Pierścieni - JRR Tolkien
- Jane Eyre - Charlott
- Harry Potter - JK Rowling
- Biblia - brak
- Wichrowe Wzgórza - Emily Bronte
- Rok 1984 - George Orwell
- Mroczne materie - Phillip Pullman
- Male kobiety - Louisa M Alcott
- Paragraf 22 - Joseph Heller
- Rebeka - Daphne Du Maurier
- Hobbit - Peter Jackson
- Buszujący w zbożu - JD Salinger



W „Wypożyczonych książkach” znajdują się pozycje, które zostały wypożyczone przez wszystkich czytelników.



W zakładce „Czytelnicy” znajdują się wszyscy zarejestrowani czytelnicy, wraz z ich loginami oraz hasłami.



Po kliknięciu w zakładkę „Informacje” ukaze nam się okno główne menu, które widzimy po zalogowaniu się do systemu.

## 6. Listing kodu

### a) C++

```
//funkcja main, wczytanie dwóch vectorów z książkami oraz czytelnikami
//wczytanie do nich książek oraz osób, a następnie wywołanie pierwszego
//okna aplikacji - WybórUzytkownika
int main(int argc, char *argv[])
{
    vector <Czytelnik> osoby;
    vector <Ksiazka> ksiazki;

    BazaKsiazek ksiazka(ksiazki);

    QApplication a(argc, argv);

    WybórUzytkownika w;

    plik d(osoby);

    w.show();

    return a.exec();
}

//konstruktor dodający obecnego czytelnika do pliku, z którego korzysta
//przez cały czas używania systemu
AktualnyCzytelnik::AktualnyCzytelnik(string im, string naz, string log,
string has)
    : imie(im), nazwisko(naz), login(log), haslo(has)
{
    ofstream obecnaOsoba("E:\\Projekt\\build-SystemBiblioteki-
Desktop_Qt_5_8_0_MinGW_32bit-Debug\\debug\\obecnyCzytelnik");

    if(obecnaOsoba.good() == false)
    {
        qApp->quit();
    }

    obecnaOsoba << imie << " " << nazwisko << " " << login << " " << haslo;

    obecnaOsoba.close();
}

//zatwierdzenie loginu i hasła dla bibliotekarza oraz sprawdzenie czy są
//poprawne, login = admin, hasło = admin
void LogowanieBibliotekarz::on_pushButton_clicked()
{
    if(ui->Admin_Login->text() == "admin" && ui->Admin_Haslo->text() ==
"admin" )
    {
        BibliotekaBibliotekarz *logBibl;
        logBibl = new BibliotekaBibliotekarz();

        hide();
        logBibl->show();
    }
}
```

```

    }
    else
    {
        QMessageBox::information(this, "title2", "bledny login lub haslo");
    }
}

//zalogowanie do systemu po wpisaniu loginu i hasla
//sprawdzenie poprawności wpisanych danych oraz dodanie aktualnego
//czytelnika poprzez zapisanie go w pliku
void LogowanieCzytelnik1::on_pushButton_clicked()
{
    BibliotekaCzytelnik *oknoCzytelnika;
    oknoCzytelnika = new BibliotekaCzytelnik();
    fstream osoba;
    string im, naz, log, has;

    osoba.open("E:\\Projekt\\build-SystemBiblioteki-
Desktop_Qt_5_8_0_MinGW_32bit-Debug\\debug\\baza.txt");

    if(osoba.good()==false)
    {
        qApp->quit();
    }

    while(!osoba.eof())
    {
        osoba >> im >> naz >> log >> has;

        if(ui->czyt_login->text().toStdString() == log && ui->czyt_haslo-
>text().toStdString() == has)
        {
            AktualnyCzytelnik(im, naz, log, has); //zapisane danych
zalogowanego uzytkownika w pliku
            hide();
            oknoCzytelnika->show(); //zalogowanie sie czytelnika
posiadajacego konto do bazy
            osoba.close();
        }
    }

    if(ui->czyt_login->text().toStdString() != log && ui->czyt_haslo-
>text().toStdString() != has)
    {
        osoba.close();
        QMessageBox::information(this, "Logowanie", "Błędny login lub
hasło, sprawdź czy na"
                                "pewno posiadasz
konto");
        LogowanieCzytelnik logowCzyt;
        hide();
        logowCzyt.setModal(true); //ponowne otwarcie konta po podanych
blednym loginie lub hasle
        logowCzyt.exec();
    }
}

//rejestracja czytelnika, dodanie czytelnika do wektora przechowujacego
//czytelnikow oraz dodanie go do pliku przechowujacego czytelnikow
//z danymi wpisanymi do textbxa

```

```

void LogowanieCzytelnik2::on_pushButton_clicked()
{
    Czytelnik kolejny;
    fstream baza;
    vector <Czytelnik> lista;
    baza.open("E:\\Projekt\\build-SystemBiblioteki-
Desktop_Qt_5_8_0_MinGW_32bit-Debug\\debug\\baza.txt", ios::app);

    if(baza.good() == false)
    {
        QApplication->quit();
    }

    kolejny.imie = ui->podajImie->text().toString();
    kolejny.nazwisko = ui->podajNazwisko->text().toString();
    kolejny.login = ui->podajLogin->text().toString();
    kolejny.haslo = ui->podajHaslo->text().toString();

    baza << endl << kolejny.imie << " " << kolejny.nazwisko << " " <<
kolejny.login << " " << kolejny.haslo;

    lista.push_back(kolejny);

    baza.close();

    QMessageBox::information(this, "Poprawna rejestracja", "Rejestracja
przebiegła pomyślnie, zaloguj się do systemu");

    LogowanieCzytelnik logowCzyt;
    hide();
    logowCzyt.setModal(true); //ponowne otworenie konta po poprawnej
rejestracji
    logowCzyt.exec();
}

//wczytanie czytelnikow z pliku do okna aplikacji wyświetlającego w menu
//bibliotekarza wszystkich zarejestrowanych osób
ObecniCzytelnicy::ObecniCzytelnicy(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::ObecniCzytelnicy)
{
    ui->setupUi(this);

    fstream baza_osob;

    baza_osob.open("E:\\Projekt\\build-SystemBiblioteki-
Desktop_Qt_5_8_0_MinGW_32bit-Debug\\debug\\baza.txt", ios::in);

    if(baza_osob.good() == false)
    {
        QApplication->quit();
    }

    string imieCzyt, nazwiskoCzyt, a, b;

    while(!baza_osob.eof())
    {
        baza_osob >> imieCzyt >> nazwiskoCzyt >> a >> b;

        QString qimieCzyt = QString::fromStdString(imieCzyt);
        QString qnazwiskoCzyt = QString::fromStdString(nazwiskoCzyt);
    }
}

```

```

        ui->listaObecnnychCzytelnikow->addItem(qimieCzyt + " " +
qnazwiskoCzyt);
    }

    baza_osob.close();
}

//wczytanie książek do okna wyświetlającego dostępne książki
PrzegladaJKsiazki::PrzegladaJKsiazki(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::PrzegladaJKsiazki)
{
    ui->setupUi(this);

    vector <Ksiazka> ksiazka;
    fstream plik;

    plik.open("E:\\Projekt\\build-SystemBiblioteki-
Desktop_Qt_5_8_0_MinGW_32bit-Debug\\debug\\plikKsiazki");

    if(plik.good() == false)
    {
        qApp->quit();
    }

    while(!plik.eof())
    {
        Ksiazka nowa;

        getline(plik, nowa.tytul);
        getline(plik, nowa.autor);
        getline(plik, nowa.data_wyp);
        getline(plik, nowa.data_zwrotu);

        ksiazka.push_back(nowa);
    }
    //jeśli książka w pliku ma date ustawiona jako „na stanie” to następuje
    //jej wczytanie i wyświetlenie w oknie aplikacji
    int elementy = ksiazka.size();

    for(int i = 0; i < elementy; i++)
    {
        if(ksiazka[i].data_wyp == "na stanie")
        {
            string tytul = ksiazka[i].tytul;
            string autor = ksiazka[i].autor;
            QString qtytul = QString::fromStdString(tytul);
            QString qautor = QString::fromStdString(autor);
            ui->listaDostepnychKsiazek->addItem(qtytul + " - " + qautor);
        }
    }
    plik.close();
}

```

```

//zarezerwowanie wybranej ksiazki
void PrzegladaJksiazki::on_PrzyciskRezerwuj_clicked()
{
    fstream ksiazki;
    fstream obecnaOsoba;
    fstream wypozyzione;

    obecnaOsoba.open("E:\\Projekt\\build-SystemBiblioteki-
Desktop_Qt_5_8_0_MinGW_32bit-Debug\\debug\\obecnyCzytelnik", ios::in);

    if(obecnaOsoba.good() == false)
    {
        QApplication->quit();
    }

    string imie, nazwisko;
    QString wybrane;
    //wczytanie danych osoby probujacej przedluzyc zwrot
    obecnaOsoba >> imie >> nazwisko;

    wypozyzione.open("E:\\Projekt\\build-SystemBiblioteki-
Desktop_Qt_5_8_0_MinGW_32bit-Debug\\debug\\wypozyczenia\\"
                    + imie + nazwisko + ".txt", fstream::app);

    if(wypozyzione.good() == false)
    {
        QApplication->quit();
    }

    //wczytanie ksiazek z okna aplikacji i wyszukanie interesujacej nas pozycji
    foreach(QListWidgetItem *ksiazka, ui->listaDostepnychKsiazek-
>selectedItems())
        wybrane = ksiazka->text();

    string wypozyczonaKsiazka;
    string dataWypozyczenia = date();
    string dataOddania = data_oddania();

    wypozyczonaKsiazka = wybrane.toStdString();

    ksiazki.open("E:\\Projekt\\build-SystemBiblioteki-
Desktop_Qt_5_8_0_MinGW_32bit-Debug\\debug\\plikKsiazki", ios::in);

    if(ksiazki.good() == false)
    {
        QApplication->quit();
    }

    int i = 0;
    //wyszukanie w pliku wybranej przez nas ksiazki i wczytanie jej nazwy
    //oraz tytułu do zmiennych
    while(!ksiazki.eof())
    {
        string a, b;
        getline(ksiazki, a);
        getline(ksiazki, b);
        i+=2;
        string calosc = a + " - " + b;

        if(strcmp(calosc.c_str(), wypozyczonaKsiazka.c_str()) == 0)
        {
            fstream temp;

```

```

        temp.open("E:\\Projekt\\build-SystemBiblioteki-
Desktop_Qt_5_8_0_MinGW_32bit-Debug\\debug\\temp", ios::app);
        if(temp.good() == false)
        {
            QApplication->quit();
        }
//ustawienie wskaźnika w pliku na początek a następnie przekopiowanie
//zawartości pliku do pliku tymczasowego
        ksiazki.seekg(0, ios_base::beg);

        for(int j=1; j <= i; j++)
        {
            string wiersz;
            getline(ksiazki, wiersz);
            temp.write(&wiersz[0], wiersz.length());
            temp.write("\n", 1);
        }
//zmiana daty oddania na ta na która chcemy przedluzyc nasze wypożyczenia
//w pliku tymczasowym
        string temp1, temp2;
        temp.write(&dataWypozyczenia[0], dataWypozyczenia.length());
        temp.write("\n", 1);
        getline(ksiazki, temp1);
        temp.write(&dataOddania[0], dataOddania.length());
        getline(ksiazki, temp2);
        temp.write("\n", 1);

        while(!ksiazki.eof())
        {
            string wiersz;
            getline(ksiazki, wiersz);
            temp.write(&wiersz[0], wiersz.length());
            temp.write("\n", 1);
        }
        temp.close();
        break;
    }
}
ksiazki.close();
//usuniecie starego pliku z książkami, zmiana nazwy pliku tymczasowego ze
//zmienionymi danymi oraz zmiana jego nazwy na nazwe starego pliku

        remove("E:\\Projekt\\build-SystemBiblioteki-
Desktop_Qt_5_8_0_MinGW_32bit-Debug\\debug\\plikKsiazki");

        rename("E:\\Projekt\\build-SystemBiblioteki-
Desktop_Qt_5_8_0_MinGW_32bit-Debug\\debug\\temp", "E:\\Projekt\\build-
SystemBiblioteki-Desktop_Qt_5_8_0_MinGW_32bit-Debug\\debug\\plikKsiazki");

        wypozyzione << wypozyczonaKsiazka << endl << dataWypozyczenia << endl
<< dataOddania << endl;

        qDeleteAll(ui->listaDostepnychKsiazek->selectedItems());

        obecnaOsoba.close();
        wypozyzione.close();

        //QMessageBox::information(this, "Status", "Udało się wypożyczyć
książkę");
        //hide();
    }
}

```

```

//funkcja wyszukująca książkę na liście (prosta wyszukiwarka, żeby nie
//trzeba było scrollować i szukać, przydatne przy dużej ilości książek
void PrzegladajKsiazki::on_przyciskWyszukaj_clicked()
{
    QString wyszukajTekst = ui->lineWyszukaj->text();
    int rozmiarListy = ui->listaDostepnychKsiazek->count();

    for (int k = 0; k < rozmiarListy; k++)
    {
        if (ui->listaDostepnychKsiazek->item(k)-
>text().startsWith(wyszukajTekst))
        {
            ui->listaDostepnychKsiazek->item(k)->setHidden(false);
        }
        else
        {
            ui->listaDostepnychKsiazek->item(k)->setHidden(true);
        }
    }
}

//wyświetlenie wypożyczonych przez nas pozycji na liście w oknie
//"Wypożyczone książki"
wypozytczoneKsiazki::wypozytczoneKsiazki(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::wypozytczoneKsiazki)
{
    ui->setupUi(this);

    fstream czytelnik;
    fstream wypozytczone;

    czytelnik.open("E:\\Projekt\\build-SystemBiblioteki-
Desktop_Qt_5_8_0_MinGW_32bit-Debug\\debug\\obecnyCzytelnik", ios::in);

    if(czytelnik.good() == false)
    {
        QApplication->quit();
    }
//wczytanie imienia i nazwiska osoby wypożyczającej
    string nazwa, imie, nazwisko;
    czytelnik >> imie >> nazwisko;
    czytelnik.close();

    nazwa = imie + nazwisko;

    wypozytczone.open("E:\\Projekt\\build-SystemBiblioteki-
Desktop_Qt_5_8_0_MinGW_32bit-Debug\\debug\\wypozytczenia\\" + nazwa +
".txt", ios::in);

    string tytul, data_wyp, data_odd, a, c;
//przeszukanie pliku z wypożyczeniami i wyświetlenie książek wraz z ich
//parametrami jak data tytul itp.
    wypozytczone.seekg(-2, ios_base::end);
    streampos poz = wypozytczone.tellg();
    wypozytczone.seekg(0, ios_base::beg);

    while(wypozytczone.tellg() < poz)
    {
        getline(wypozytczone, tytul);
    }
}

```



```

        getline(wypozyzione, data_wyp);
        getline(wypozyzione, data_odd);
        QString qtytul = QString::fromStdString(tytul);
        QString qdata_wyp = QString::fromStdString(data_wyp);
        QString qdata_odd = QString::fromStdString(data_odd);
        ui->listaWypozyczonychKsiazek->addItem(qtytul + " Od: " + qdata_wyp
+ " Do: " + qdata_odd);
    }
    wypozyzione.close();
}

```

## b) C#

//funkcja main, nie zostają w niej wywoływane żadne funkcje poza startem pierwszego  
//okienka programu WyborUzytkownika, nie sa inicjalizowane tez żadne listy  
//np. lista książek bądź czytelników, wszystko dzieje się w klasie Repository  
//inicjalizują się w niej obie te listy i sa widoczne przez cały czas pracy aplikacji  
namespace SystemBiblioteki

```

{
    static class Program
    {
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new WyborUzytkownika());
        }
    }
}

```

//główna różnicą pomiędzy C++ a C# jest sposób w jaki korzysta się w nich z plików  
//w C# używa się do tego osobnych klas do wczytywania jak i zapisywania  
//każda operacja jest w zasadzie już zaprogramowana i wystarczy ją znaleźć  
//w C++ wszystkie operacje przeszukiwania plików musimy wykonywać sami pisząc  
//własne funkcje

```

public class BazaCzytelnikow
{
    public static List<Czytelnik> czytelnicy = new List<Czytelnik>();

    public BazaCzytelnikow()
    {
        //tworzenie bazy wszystkich obecnych czytelnikow

        FileStream plikCzytelnicy = new FileStream("E:\\Projekt
C#\\SystemBiblioteki\\SystemBiblioteki\\pliki\\plikCzytelnicy.txt", FileMode.Open,
FileAccess.Read);
        StreamReader odczyt = new StreamReader(plikCzytelnicy);

        try
        {
            while (!odczyt.EndOfStream)
            {
                Czytelnik czyt = new Czytelnik();

                czyt.imie = odczyt.ReadLine();
                czyt.nazwisko = odczyt.ReadLine();
                czyt.login = odczyt.ReadLine();
                czyt.haslo = odczyt.ReadLine();
            }
        }
    }
}

```

```

        czytelnicy.Add(czyt);
    }
}
catch(Exception wyjatek)
{
    MessageBox.Show(wyjatek.ToString());
    Application.Exit();
}
finally
{
    odczyt.Close();
}
}
}

```

//kolejną różnicą jest korzystanie z daty, w C# możemy skorzystać z prostej klasy  
 //DateTime, która daje nam dostęp do funkcji robiących w zasadzie wszystko  
 //możemy wczytywać sam dzień, sam miesiąc, rok, lub wszystko na raz w kilku różnych  
 //formatach, w C++ nie ma tej możliwości i działanie na dacie jest uciążliwe

```

public Data()
{
    DateTime data = DateTime.Now;

    this.dataWypozyczenia = (data.Day.ToString() + "." + data.Month.ToString()
+ "." + data.Year.ToString());

    if(data.Month.ToString().Equals("12"))
    {
        this.dataOddania = (data.Day.ToString() + ".1." + (data.Year + 1));
    }
    else
    {
        this.dataOddania = (data.Day.ToString() + "." + (data.Month + 1) + "."
+ data.Year.ToString());
    }
}

```

//do korzystania z danych zawartych w klasie AktualnyCzytelnik musiałem stworzyć  
 //dodatkowa klasa mająca jeden obiekt statyczny będący obiektem właśnie tej klasy i  
 //mający wszystkie jej właściwości, dzięki temu że jest statyczny można było go używać  
 //bez wywoływania go na konkretnym obiekcie

```

public class Obsluga_AktualnyCzytelnik
{
    public static AktualnyCzytelnik zalogowany;
}

```

//roznice tez są w niektórych innych funkcjach jak np. sprawdzenie czy wpisany string  
 //nie jest pusty, wystarczy do tego proste IsNullOrEmpty, natomiast w C++  
 //należało napisać do samemu funkcje sprawdzającą ten warunek

```

private void button1_Click_1(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textBox_imie.Text) ||
string.IsNullOrEmpty(textBox_nazwisko.Text)
    || string.IsNullOrEmpty(textBox_login.Text) ||
string.IsNullOrEmpty(textBox_haslo.Text))
    {
        MessageBox.Show("Błędne dane logowania. Spróbuj ponownie");
    }
}

```

```

else
{
    Czytelnik nowy = new Czytelnik(textBox_imie.Text.ToString(),
textBox_nazwisko.Text.ToString(), textBox_login.Text.ToString(),
textBox_haslo.Text.ToString());
    BazaCzytelnikow.czytelnicy.Add(nowy);

    MessageBox.Show("Rejestracja przebiegła pomyślnie. Zaloguj się do
systemu.");
    this.Close();
    ObslugaForm.formaLogowanie.Show();
}
}

```

## 7. Wnioski

W programie „system biblioteki” mamy możliwość stworzenia własnego konta użytkownika, bądź użytkownika go jako bibliotekarza. Możemy przeglądać książki, wypożyczać je, przedłużać ich zwrot oraz sprawdzać swoje dane. Wersja dla bibliotekarza daje nam opcje przeglądania wypożyczonych książek oraz zarejestrowanych czytelników. W programie na pewno dodałbym możliwość zwrotu książki, algorytm sprawdzający, czy dany czytelnik nie przekroczył czasu wypożyczenia z nałożeniem np. opłaty za każdy dzień zwłoki. Program wysyłałby powiadomienie do użytkownika po zalogowaniu, albo blokował możliwość korzystania z systemu do czasu uregulowania zapłaty itp. Podobną funkcję możnaby dodać bibliotekarzowi, aby mógł zarządzać systemem oraz wypożyczeniami samodzielnie.

Głównym wnioskiem z realizacji projektu są na pewno różnice widoczne z pisania go w QtCreatorze przy użyciu C++ oraz w WindowForms w języku C#. Visual studio daje możliwość dużo łatwiejszego zrobienia ładnego i przejrzystego GUI, niż Qt, co widać po jakości programów w jednym jak i w drugim. Sporą różnicę też można dostrzec w samej składni języka i jego możliwościach pracy np. na plikach. C# daje nam duże bezpieczeństwo pracy na nich, a przy okazji jest to bardzo łatwe. W C++ jest to dosyć żmudne i można w prosty sposób popsuć całą swoją pracę.

Program w C++, z racji tego, że był pisany pierwszy (do tego był moim pierwszym projektem), jest dość kiepski. Większość rzeczy oprogramowywałem w funkcjach „wciśnięcia przycisku”, zamiast jako funkcje klas, z późniejszym wywołaniem ich. Podobna sytuacja miała miejsce z tworzeniem vectorów przechowujących książki i czytelników. Zamiast stworzyć klasę, inicjalizującą te vectory na starcie programu, ja ciągle wczytywałem je z pliku, co powodowało wiele błędów. Poza tym, że są w nim użyte klasy do przechowywania danych to obiektowości w nim prawie nie ma. Z kolei w C# ucząc się na błędach poprawiłem wszystkie te rzeczy. Stworzyłem w klasach BazaCzytelnikow oraz BazaKsiazek konstruktory inicjalizujące na początku programu dwie listy poprzez wczytanie z pliku do nich danych. Dzięki temu uniknąłem ciągłej pracy na plikach. Wszystkie funkcje (z wyjątkiem tych kilkunastu, które były nieopłacalne) zaprogramowałem jako metody klas, przez co teraz wystarczy je wywołać, aby spełniały swoje zadanie. Każda klasa posiada konstruktor inicjalizujący jej obiekty. Są użyte obiekty statyczne oraz bloki wyłapywania wyjątków, aby uchronić program przed „wysypywaniem się”. Również GUI jest dużo ładniejsze i program po dopracowaniu mógłby stać się funkcjonalny i użytkowy.

