Dokumentacja Systemu Bibliotecznego

Autorzy: **Magdalena Zawada** oraz **Mateusz Zawiliński,** lab3/2/AD

Wstęp

System biblioteczny to projekt zrealizowany w ramach przedmiotu "Programowanie w środowisku ASP.NET" na WSEI. Celem systemu jest umożliwienie zarządzania zbiorami bibliotecznymi, obsługi użytkowników, wypożyczeń oraz zwrotów książek. Pani bibliotekarka może przed zalogowaniem podejrzeć jedynie politykę prywatności i stronę główną, ale po zalogowaniu może rozpocząć pracę w systemie.

Dane

Platforma: ASP.NET Core

Środowisko: Visual Studio 2022

Wersja .NET: .NET 8.0 lub nowsza

Baza danych: SQL Server

Link: https://github.com/Magudarena/Biblioteka.git

Cel projektu

System biblioteczny został zaprojektowany z myślą o automatyzacji procesów związanych z zarządzaniem zasobami bibliotecznymi, takimi jak książki, użytkownicy oraz wypożyczenia. Aplikacja umożliwia zarządzanie książkami, dodawanie nowych egzemplarzy, ich edycję i usuwanie, a także obsługę użytkowników (np. rejestracja, logowanie). Dodatkowo, system obsługuje wypożyczenia książek, zarządza terminami zwrotów oraz generuje raporty.

Wymagania

Aby poprawnie uruchomić i korzystać z systemu, wymagane są następujące komponenty:

- Visual Studio z zainstalowanymi dodatkami do programowania w ASP.NET oraz odpowiednie pakiety Entity Framework – Core i SQL server, wersja projektu: .NET 8
- SQL Server lub kompatybilna baza danych, na której postawimy bazę z README.md
- Repozytorium pobrane z https://github.com/Magudarena/Biblioteka.git
- Komputer z systemem operacyjnym Windows
- RAM i odrobinę dostępnego miejsca na dysku.

Folder Biblioteka zawiera kod źródłowy aplikacji ASP.NET, składa się z elementów takich jak kontrolery, modele, widoki (MVC), oraz pliki konfiguracyjne.

Instrukcja instalacji

1. Najpierw należy pobrać repozytorium do swojego Visual Studio ze strony https://github.com/Magudarena/Biblioteka.git.
2. Po tym należy włączyć Microsoft SQL Server Management Studio i wkleić w niego podany poniżej kod wprowadzając kolejne fragmenty kodu:

```
--Skrypt do utworzenia bazy danych

use master;
create database Biblioteka;
 use Biblioteka;

--Utworzenie tabel w bazie

CREATE TABLE kategoria (
    id int IDENTITY(1,1) PRIMARY KEY,
    nazwa varchar(40) UNIQUE,
);

CREATE TABLE klient (
    id int IDENTITY(1,1) PRIMARY KEY,
    imie varchar(20),
    nazwisko varchar(40),
    telefon varchar(9) UNIQUE,
    email varchar(60) UNIQUE
);

CREATE TABLE ksiazka (
    id int IDENTITY(1,1) PRIMARY KEY,
    nr_biblioteczny varchar(10) UNIQUE,
    tytul varchar(100),
    autor varchar(60),
    ISBN varchar(13),
    kategoria int FOREIGN KEY REFERENCES kategoria(id),
    dostepna bit
);

CREATE TABLE uprawnienia (
    id int IDENTITY(1,1) PRIMARY KEY,
    nazwa varchar(30)
);

CREATE TABLE uzytkownicy (
    id int IDENTITY(1,1) PRIMARY KEY,
    email varchar(60) UNIQUE,
    haslo varchar(100),
    imie varchar(20),
    nazwisko varchar(40),
    id_uprawnienia int FOREIGN KEY REFERENCES uprawnienia(id)
);

CREATE TABLE wypozyczenia (
    id int IDENTITY(1,1) PRIMARY KEY,
    id_ksiazka int FOREIGN KEY REFERENCES ksiazka(id) ON DELETE CASCADE,
    id_klient int FOREIGN KEY REFERENCES klient(id) ON DELETE CASCADE,
    data_wypozyczenia datetime,
    data_zwrotu datetime
);

--Po utworzeniu bazy należy utworzyć widoki

create view KsiazkaPerKlient as
```

```
select w.id as "id_wypozyczenia", kl.id as "id_klient", ks.id as "id_ksiazka",
ks.nr_biblioteczny, ks.tytul, ks.autor, ks.ISBN, w.data_wypozyczenia, w.data_zwrotu  from
wypozyczenia w
right join klient kl ON w.id_klient = kl.id
left join ksiazka ks on w.id_ksiazka = ks.id;

Create View ListaKlientow as
select k.id, k.imie as "imie", k.nazwisko as "nazwisko", k.telefon as "telefon", k.email as
"email", (count(w.data_wypozyczenia) - count(w.data_zwrotu)) as "wypozyczone" from
wypozyczenia w
right join klient k on k.id = w.id_klient
group by k.id, k.imie, k.nazwisko, k.telefon, k.email;

create view ListaKsiazek as
select ks.id, ks.nr_biblioteczny, ks.tytul, ks.autor, ks.ISBN, kt.nazwa as "kategoria",
ks.dostepna from ksiazka ks
join kategoria kt on ks.kategoria = kt.id;

--Należy dodać poziomy uprawnień użytkowników
INSERT INTO uprawnienia (nazwa) VALUES ('Administrator');
INSERT INTO uprawnienia (nazwa) VALUES ('Lepsze');
INSERT INTO uprawnienia (nazwa) VALUES ('Gorsze');
INSERT INTO uprawnienia (nazwa) VALUES ('Nowy');
```

3. Teraz można uruchomić program. Jeżeli nie łączy się z bazą danych, to należy jedynie zmienić nazwę „localhost” w pliku appsettings,json na nazwę serwera na którym znajduje się lokalna baza.

Schemat działania

System biblioteczny realizuje następujące funkcjonalności:

1. Zarządzanie użytkownikami:
   - Rejestracja nowych użytkowników
   - Logowanie użytkowników
2. Zarządzanie książkami:
   - Dodawanie nowych książek do bazy danych
   - Edytowanie szczegółów książek (np. tytuł, autor, ISBN)
   - Usuwanie książek z systemu
3. Wypożyczenia:
   - Wypożyczenie książki przez użytkownika
   - Monitorowanie terminu zwrotu książek
   - Obsługa opóźnień i kar za nieterminowe zwroty
4. Zarządzanie klientami:
   - Dodawanie, edytowanie, usuwanie klientów
   - Wyświetlanie wypożyczonych książek
   - Wyświetlanie historii wypożyczania
5. Raportowanie:
   - Generowanie raportów o dostępności książek w bibliotece
   - Historia wypożyczania dla każdej książki
   - Przegląd wypożyczania poszczególnych użytkowników
   - Wyszukiwanie użytkowników, którzy nie oddali książek na czas

Opis funkcjonalności

Zarządzanie użytkownikami

- Rejestracja – Użytkownicy mogą tworzyć konto podając podstawowe dane.
- Logowanie – Użytkownicy mogą zalogować się, a system zarządza sesjami użytkowników.
- Edycja – Bibliotekarze mogą edytować dane użytkowników oraz klientów.
- Usuwanie klienta – Usunięcie użytkownika biblioteki z systemu.

Zarządzanie książkami

- Dodawanie – Bibliotekarze mogą dodawać nowe książki do systemu.
- Edycja – Możliwość edytowania istniejących rekordów książek.
- Usuwanie – Możliwość usuwania książek z katalogu, jeżeli nie ma już wypożyczeń.

Wypożyczenia książek

- Wypożyczenie książki – Klienci mogą wypożyczyć książki czas monitorowany przez system.
- Zwroty książek – Klienci mogą zwrócić książki, a system oblicza datę zwrotu oraz opóźnienia.

Raportowanie

- Raport o książkach – Generowanie listy dostępnych i wypożyczonych książek.
- Raport o użytkownikach – Przegląd wypożyczeń poszczególnych użytkowników.
- Raport o opóźnieniach – Wyszukiwanie użytkowników, którzy nie oddali książek na czas.

Architektura repozytorium

Projekt oparty jest na wzorcu Model-View-Controller, który dzieli aplikację na trzy główne części:

Modele

Model w tym systemie reprezentuje dane aplikacji i jest odpowiedzialny za interakcję z bazą danych. Główne modele danych:

- BibliotekaContext.cs – Klasa kontekstu bazy danych, dziedziczy po DbContext.
- ErrorViewModel.cs – Model widoku błędu do informacji o błędach w aplikacji.
- Kategoria.cs – Model reprezentujący kategorię książek w bibliotece.
- Klient.cs – Model reprezentujący klienta (użytkownika) systemu bibliotecznego.
- Ksiazka.cs – Model reprezentujący książkę w bibliotece.
- KsiazkaPerKlient.cs – Model, który reprezentuje wypożyczenie książki przez klienta.
- ListaKlientow.cs – Model wykorzystywany do prezentacji wszystkich klientów biblioteki.
- ListaKsiazek.cs – Model listy książek w celu prezentacji książek w różnych kontekstach.
- LogowanieModel.cs – Model do obsługi logowania użytkownika.
- LogowanieViewModel.cs – Model do obsługi logowania użytkownika.
- RegisterViewModel.cs – Model zawierający dane potrzebne do rejestracji użytkownika.
- Uprawnienia.cs – Model zajmujący się uprawnieniami naszych użytkowników.
- Uzytkownik.cs – Model reprezentujący użytkownika systemu (bibliotekarza).
- Wypozyczenie.cs – Model reprezentujący wypożyczenie książki przez użytkownika.
- Zwracanie.cs – Model odpowiadający za zwrot książki w systemie bibliotecznym.

Model danych jest powiązany z Entity Framework Core, co pozwala na mapowanie obiektów na tabele w bazie danych.

Widoki

Widoki są odpowiedzialne za wyświetlanie danych użytkownikowi. W tym projekcie wykorzystywane są Razor Views, które pozwalają na dynamiczne generowanie HTML z osadzonymi fragmentami C#. Widoki znajdują się w folderze Views.

1. Home
   - Index.cshtml – Strona główna aplikacji.
   - Privacy.cshtml – Strona zawierająca informacje o polityce prywatności.
2. Kategorie
   - Dodaj.cshtml – Formularz dodawania nowej kategorii książek.
   - Kategorie.cshtml – Widok pokazujący dostępne kategorie książek.
   - Usun.cshtml – Formularz usuwania kategorii książek.
3. Klienci
   - Edytuj.cshtml – Formularz edycji danych klienta.
   - KsiazkiKlienta.cshtml – Widok przedstawiający książki wypożyczone przez klienta.
   - ListaKlientow.cshtml – Lista wszystkich klientów w systemie.
   - NowyKlient.cshtml – Formularz dodawania nowego klienta.
   - Usun.cshtml – Formularz usuwania klienta.
4. Ksiazki
   - Edytuj.cshtml – Formularz edycji danych książki.
   - ListaKsiazek.cshtml – Lista książek w bibliotece.
   - NowaKsiazka.cshtml – Formularz dodawania nowej książki.
   - Usun.cshtml – Formularz usuwania książki.
   - Zwroc.cshtml – Formularz zwrotu książki.
5. Logowanie
   - Logowanie.cshtml – Widok odpowiadający za formularz logowania użytkownika.
6. Rejestracja
   - Rejestracja.cshtml – Formularz rejestracji nowego użytkownika.
7. Shared
   - Error.cshtml – Strona błędu, wyświetlana w przypadku problemów z aplikacją.
   - _Layout.cshtml – Globalny layout dla aplikacji, nagłówek, stopkę i elementy interfejsu.
   - _Layout.cshtml.css – Arkusz stylów dla layoutu aplikacji.
   - _ValidationScriptsPartial.cshtml – Częściowa strona skryptów do walidacji formularzy.
8. Uzytkownicy
   - Uzytkownicy.cshtml –Strona z listą użytkowników (bibliotekarek) z ich uprawnieniami.
9. Wypozyczenia
   - BladKlienta.cshtml – Strona informująca o błędzie związanym z klientem.
   - PodsumowanieKsiazki.cshtml – Widok podsumowujący książkę podczas wypożyczenia.
   - WybierzKlienta.cshtml – Formularz wyboru klienta przy wypożyczeniu książki.
   - Wypozyczenia.cshtml – Formularz wypożyczenia książki przez użytkownika.
10. Zwracanie
    - Zwracanie.cshtml – Formularz inicjujący proces zwrotu książki.
    - Zwroc.cshtml – Strona do zakończenia procesu zwrotu książki.

Kontrolery

Kontroler to warstwa, która pośredniczy pomiędzy użytkownikiem a danymi. Odpowiada za obsługę żądań HTTP i komunikację z modelem, a także przekazywanie danych do widoków. W projekcie znajdują się kontrolery takie jak:

- HomeController.cs – kontroler obsługujący logikę strony głównej.
- KategorieController.cs – kontroler odpowiedzialny za operacje z kategoriami książek.
- KlienciController.cs – kontroler do zarządzania klientami i użytkownikami systemu.
- KsiazkaController.cs – kontroler do operacji na książkach (np. dodawanie, edytowanie).
- LogowanieController.cs – kontroler do działań związanych z logowaniem użytkowników.
- RejestracjaController.cs – kontroler do działań związanych z rejestracją użytkowników.
- UzytkownicyController.cs – kontroler do zarządzania kontami użytkowników przez admina.
- WypozyczeniaController.cs – kontroler obsługujący operacje związane z wypożyczeniami.
- ZwracanieController.cs – kontroler do obsługi procesu zwrotu książek.

Baza danych

Baza danych Biblioteka została zaprojektowana do zarządzania zasobami bibliotecznymi, takimi jak książki, klienci, wypożyczenia oraz użytkownicy systemu. Baza danych jest zaprojektowana pod względem integralności danych, wykorzystując klucze obce do utrzymania spójności między tabelami (np. książki są przypisane do kategorii, wypożyczenia są związane z książkami i klientami). Ponadto, tabeli uzytkownicy przypisano odpowiednie poziomy uprawnień, co zapewnia kontrolę dostępu do danych. Oto szczegółowa analiza struktury bazy:

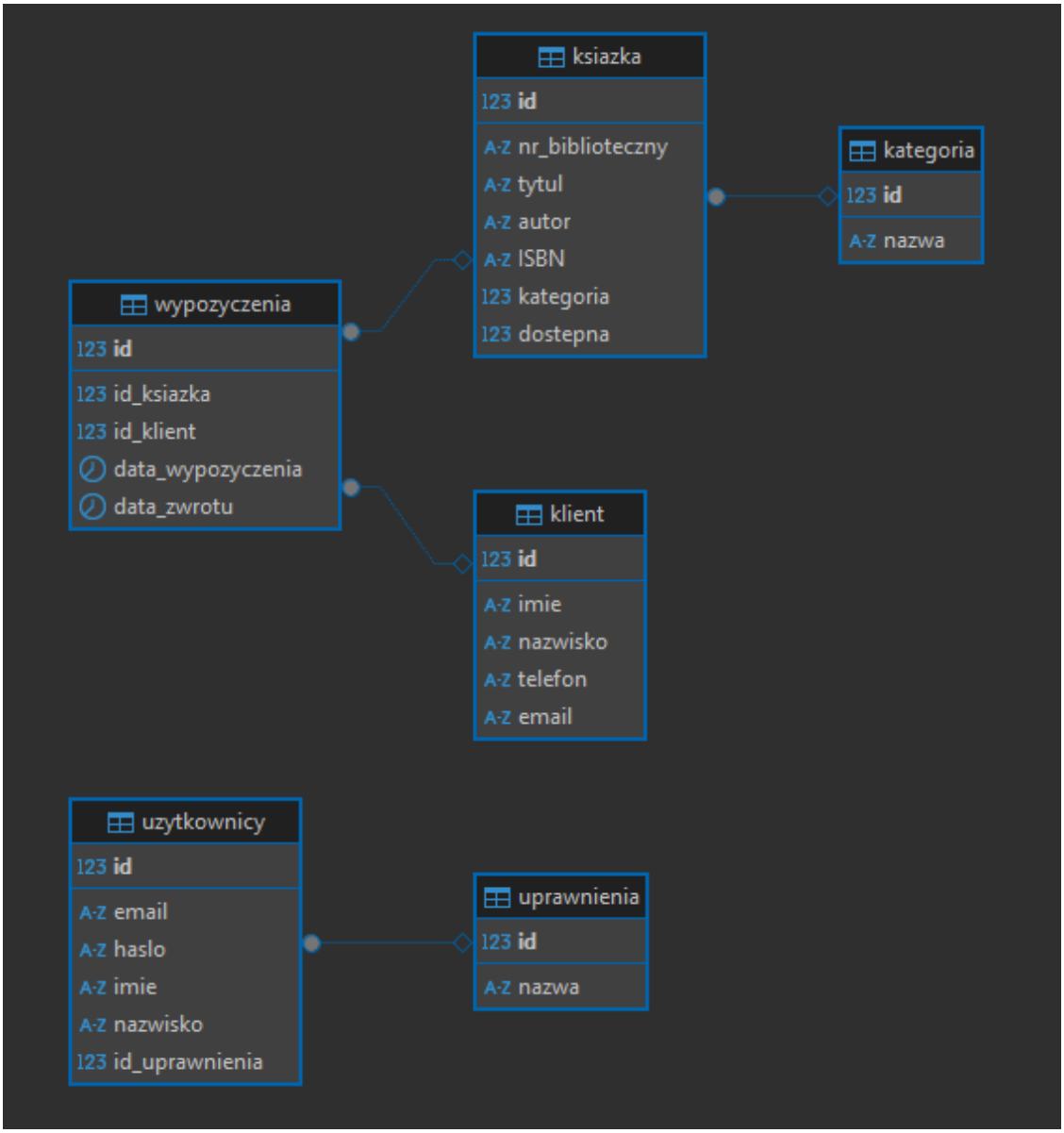
Tabele w bazie danych

- kategoria – Przechowuje informacje o kategoriach książek w bibliotece, umożliwiając przypisanie książek do odpowiednich grup tematycznych.
- klient – Zawiera dane klientów wypożyczających książki, takie jak imię, nazwisko, numer telefonu i adres e-mail.
- ksiazka – Przechowuje dane o książkach w bibliotece, w tym tytuł, autor, ISBN oraz przypisanie do kategorii książki. Zawiera także informację o dostępności książki.
- uprawnienia – Określa poziomy uprawnień dla użytkowników systemu, takie jak administratorzy czy standardowi użytkownicy.
- uzytkownicy – Zawiera dane użytkowników systemu, w tym dane logowania (e-mail, hasło) oraz przypisanie do poziomu uprawnień z tabeli uprawnienia.
- wypozyczenia – Rejestruje wypożyczenia książek przez klientów, zawierając daty wypożyczenia oraz zwrotu, a także powiązanie z książką i klientem.

Widoki w bazie danych

- KsiazkaPerKlient – Wyświetla informacje o wypożyczeniach książek przez klientów, w tym numery katalogowe książek, tytuły oraz daty wypożyczenia i zwrotu.
- ListaKlientow – Pokazuje listę wszystkich klientów, ich dane kontaktowe oraz liczbę książek aktualnie wypożyczonych (nieoddanych).
- ListaKsiazek – Prezentuje wszystkie książki w bibliotece, zawierając szczegóły dotyczące tytułów, autorów, ISBN, kategorii oraz dostępności książek.

Wizualizacja bazy danych



Procesy

System zarządza wieloma procesami, które są realizowane w sposób sekwencyjny:

- Rejestracja i logowanie użytkowników: Po rejestracji użytkownik może się zalogować, a system przydziela mu podstawową rolę, którą później może zmienić administrator z najwyższymi uprawnieniami (np. bibliotekarka, superadmin bibliotekarka).
- Zarządzanie książkami: Bibliotekarze mogą dodawać nowe książki do systemu, edytować istniejące lub je usuwać.
- Wypożyczenia książek: Użytkownicy mogą wypożyczać książki klientom na określony czas. System śledzi daty wypożyczeń i generuje przypomnienia o zwrotach.
- Raportowanie: System generuje raporty o książkach, wypożyczeniach i opóźnieniach, które mogą być pomocne dla bibliotekarzy w zarządzaniu biblioteką.

Zabezpieczenia

1. W projekcie kontroler korzysta z bibliotek związanych z autentykacją, takich jak:
   - Microsoft.AspNetCore.Authentication – do zarządzania procesem autentykacji.
   - Microsoft.AspNetCore.Authentication.Cookies – do obsługi cookies sesyjnych, które są używane do przechowywania informacji o zalogowanych użytkownikach.
   - Microsoft.AspNetCore.Identity – do integracji z ASP.NET Identity, który jest rozwiązaniem do zarządzania użytkownikami i ich rolami.

2. Metoda Logowanie(LogowanieViewModel model):
   Metoda ta obsługuje POST, sprawdzając dane logowania wprowadzane przez użytkownika. Używa modelu LogowanieViewModel, który zawiera dane takie jak e-mail i hasło.

3. Wyszukiwanie użytkownika w bazie danych:
   W metodzie Logowanie(LogowanieViewModel model) wykonywane jest wyszukiwanie użytkownika w bazie danych na podstawie adresu e-mail.

4. Ochrona przed atakami CSRF:
   W metodzie POST zastosowaliśmy atrybut [ValidateAntiForgeryToken], co chroni aplikację przed atakami typu Cross-Site Request Forgery (CSRF).

5. Haszowanie haseł:
   Hasła użytkowników są haszowane przed zapisaniem w bazie danych, dzięki czemu nie są przechowywane w postaci jawnej. Używany jest mechanizm PasswordHasher<T>, który zapewnia bezpieczne haszowanie oraz weryfikację haseł.

6. Autentykacja oparta na cookies:
   Po pomyślnym logowaniu użytkownik otrzymuje ciasteczka sesyjne, które przechowują dane o sesji. Ciasteczka są bezpieczne, gdyż można je skonfigurować z flagami Secure i HttpOnly, co zabezpiecza je przed atakami XSS.

7. Ochrona przed SQL Injection:
   Dzięki wykorzystaniu Entity Framework oraz zapytań parametryzowanych, aplikacja jest odporna na ataki SQL Injection.

Przykładowe dane

Celem przetestowania działania aplikacji przygotowaliśmy przykładowe treści w skryptach gotowych do zaimplementowania w bazie danych:

```
--Hasła testowe: Test,1
--przykłady użytkowników
INSERT INTO uzytkownicy (email, haslo, imie, nazwisko, id_uprawnienia)
VALUES
('superadmin@example.com',
'AQAAAAIAAYagAAAAEO2rrTz9lR5MmUbI3dz1BjQmjVmk0fzyvJri2sycGyizeRqxLr+1kLj1xuvfZfPsPg==', 'Super',
'Admin', 1),
('bibliotekarka@example.com',
'AQAAAAIAAYagAAAAEO2rrTz9lR5MmUbI3dz1BjQmjVmk0fzyvJri2sycGyizeRqxLr+1kLj1xuvfZfPsPg==', 'Anna',
'Bibliotekarka', 2),
('sambz@mail.pl',
'AQAAAAIAAYagAAAAEO2rrTz9lR5MmUbI3dz1BjQmjVmk0fzyvJri2sycGyizeRqxLr+1kLj1xuvfZfPsPg==',
'Sameligelimelion', 'Brzęczyszczykiewicz', 4);
```

```
--przykłady kategorii
insert into kategoria (nazwa) values ('Biografie');
insert into kategoria (nazwa) values ('Dla dzieci');
insert into kategoria (nazwa) values ('Dla młodzieży');
insert into kategoria (nazwa) values ('Fantastyka, horror');
insert into kategoria (nazwa) values ('Historia');
insert into kategoria (nazwa) values ('Komiks');
insert into kategoria (nazwa) values ('Kryminał, Sensacja, Thriller');
insert into kategoria (nazwa) values ('Literatura faktu');
insert into kategoria (nazwa) values ('Literatura obyczajowa');
insert into kategoria (nazwa) values ('Literatura piękna');
insert into kategoria (nazwa) values ('Podręczniki');

--przykłady książek
insert into ksiazka (nr_biblioteczny, tytul, autor, ISBN, kategoria, dostepna) values
('0000000001', 'Steve Jobs', 'Walter Isaacson', '9781451648539', 1, 1);
insert into ksiazka (nr_biblioteczny, tytul, autor, ISBN, kategoria, dostepna) values
('0000000002', 'Becoming. Moja historia', 'Michelle Obama', '9788381691626', 1, 0);
insert into ksiazka (nr_biblioteczny, tytul, autor, ISBN, kategoria, dostepna) values
('0000000003', 'Mały Książe', 'Antoine de Saint-Exupéry', '9788372784622', 2, 1);
insert into ksiazka (nr_biblioteczny, tytul, autor, ISBN, kategoria, dostepna) values
('0000000004', 'Opowieści z Narnii: Lew, czarownica i stara szafa', 'C.S. Lewis',
'9788372780143', 2, 1);
insert into ksiazka (nr_biblioteczny, tytul, autor, ISBN, kategoria, dostepna) values
('0000000005', 'Igrzyska śmierci', 'Suzanne Collins', '9788310126351', 3, 0);
insert into ksiazka (nr_biblioteczny, tytul, autor, ISBN, kategoria, dostepna) values
('0000000006', 'Percy Jackson i bogowie olimpijscy: Złodziej pioruna', 'Rick Riordan',
'9781423134947', 3, 0);
insert into ksiazka (nr_biblioteczny, tytul, autor, ISBN, kategoria, dostepna) values
('0000000007', 'Władca Pierścieni Drużyna Pierścienia', 'J.R.R. Tolkien', '9780007117116', 4, 0);
insert into ksiazka (nr_biblioteczny, tytul, autor, ISBN, kategoria, dostepna) values
('0000000008', 'To', 'Stephen King', '9788379857831', 4, 0);
insert into ksiazka (nr_biblioteczny, tytul, autor, ISBN, kategoria, dostepna) values
('0000000009', 'Historia Polski 1914-1939', 'Andrzej Chwalba', '9788381424002', 5, 0);
insert into ksiazka (nr_biblioteczny, tytul, autor, ISBN, kategoria, dostepna) values
('0000000010', 'Sapiens. Od zwierząt do bogów', 'Yuval Noah Harari', '9780062316097', 5, 0);
insert into ksiazka (nr_biblioteczny, tytul, autor, ISBN, kategoria, dostepna) values
('0000000011', 'Batman: Rok pierwszy', 'Frank Miller', '9781401207526', 6, 1);
insert into ksiazka (nr_biblioteczny, tytul, autor, ISBN, kategoria, dostepna) values
('0000000012', 'Maus', 'Art Spiegelman', '9780141014081', 6, 1);
insert into ksiazka (nr_biblioteczny, tytul, autor, ISBN, kategoria, dostepna) values
('0000000013', 'Dziewczyna z pociągu', 'Paula Hawkins', '9780593072493', 7, 1);
insert into ksiazka (nr_biblioteczny, tytul, autor, ISBN, kategoria, dostepna) values
('0000000014', 'Zaginiona dziewczyna', 'Gillian Flynn', '9780297859383', 7, 1);
insert into ksiazka (nr_biblioteczny, tytul, autor, ISBN, kategoria, dostepna) values
('0000000015', 'Człowiek włóczył się po Londynie', 'Laurent Keksik', '9788328055047', 8, 1);
insert into ksiazka (nr_biblioteczny, tytul, autor, ISBN, kategoria, dostepna) values
('0000000016', 'Factfulness', 'Hans Rosling', '9781473637467', 8, 1);
insert into ksiazka (nr_biblioteczny, tytul, autor, ISBN, kategoria, dostepna) values
('0000000017', 'Właśnie tak!', 'Nicholas Sparks', '9780751542974', 9, 1);
insert into ksiazka (nr_biblioteczny, tytul, autor, ISBN, kategoria, dostepna) values
('0000000018', 'Dom nad rozlewiskiem', 'Małgorzata Kalicińska', '9788324141492', 9, 1);
insert into ksiazka (nr_biblioteczny, tytul, autor, ISBN, kategoria, dostepna) values
('0000000019', 'Przeminęło z wiatrem', 'Margaret Mitchell', '9788389951779', 10, 1);
insert into ksiazka (nr_biblioteczny, tytul, autor, ISBN, kategoria, dostepna) values
('0000000020', 'Sto lat samotności', 'Gabriel García Márquez', '9780141184999', 10, 1);
insert into ksiazka (nr_biblioteczny, tytul, autor, ISBN, kategoria, dostepna) values
('0000000021', 'Biologia na czasie 1', 'Andrzej Boczarowski', '9788302155053', 11, 0);
insert into ksiazka (nr_biblioteczny, tytul, autor, ISBN, kategoria, dostepna) values
('0000000022', 'Matematyka. Poznać, zrozumieć. Klasa 1', 'Praca zbiorowa', '9788326735613', 11,
1);



--przykłady klientów
```

```
insert into klient (imie, nazwisko, telefon, email) values ('Anna', 'Kowalska', '123456789',
'anna.kowalska@example.com');
insert into klient (imie, nazwisko, telefon, email) values ('Marek', 'Nowak', '987654321',
'marek.nowak@example.com');
insert into klient (imie, nazwisko, telefon, email) values ('Jan', 'Wiśniewski', '555123456',
'jan.wisniewski@example.com');
insert into klient (imie, nazwisko, telefon, email) values ('Ewa', 'Kamińska', '666234567',
'ewa.kaminska@example.com');
insert into klient (imie, nazwisko, telefon, email) values ('Piotr', 'Lewandowski', '777345678',
'piotr.lewandowski@example.com');
insert into klient (imie, nazwisko, telefon, email) values ('Katarzyna', 'Zielińska',
'888456789', 'katarzyna.zielinska@example.com');

--przykłady wypożyczeń
insert into wypozyczenia (id_ksiazka, id_klient, data_wypozyczenia, data_zwrotu) values (2, 1,
'2024-03-11 11:51:00.667', null);
insert into wypozyczenia (id_ksiazka, id_klient, data_wypozyczenia, data_zwrotu) values (5, 1,
'2024-11-11 13:51:00.667', null);
insert into wypozyczenia (id_ksiazka, id_klient, data_wypozyczenia, data_zwrotu) values (6, 1,
'2024-03-11 11:51:00.667', null);
insert into wypozyczenia (id_ksiazka, id_klient, data_wypozyczenia, data_zwrotu) values (8, 2,
'2024-11-11 13:51:00.667', '2024-11-26 13:51:00.667');
insert into wypozyczenia (id_ksiazka, id_klient, data_wypozyczenia, data_zwrotu) values (7, 3,
'2024-03-11 11:51:00.667', null);
insert into wypozyczenia (id_ksiazka, id_klient, data_wypozyczenia, data_zwrotu) values (10, 3,
'2024-11-11 13:51:00.667', null);
insert into wypozyczenia (id_ksiazka, id_klient, data_wypozyczenia, data_zwrotu) values (21, 4,
'2024-03-11 11:51:00.667', null);
insert into wypozyczenia (id_ksiazka, id_klient, data_wypozyczenia, data_zwrotu) values (9, 3,
'2024-11-11 13:51:00.667', '2024-11-26 13:51:00.667');
```

Potencjalny rozwój

Choć system już teraz spełnia swoje zadanie, istnieje kilka obszarów, które mogą zostać rozbudowane:

- System rekomendacji książek – na podstawie historii wypożyczeń użytkownika system mógłby sugerować wypożyczane w przyszłości książki, które zainteresują klienta.
- Integracja z systemami zewnętrznymi – połączenie z systemami katalogów książek, np. z bibliotekami cyfrowymi, które umożliwiłyby dodawanie nowych pozycji do bazy książek.
- Zaawansowana analityka i raportowanie – rozbudowa generowania raportów o popularności książek, historii wypożyczeń w danym okresie i analiza opóźnionych zwrotów książek.

Podsumowanie

System biblioteczny w technologii ASP.NET Core oparty na architekturze MVC skutecznie separuje logikę aplikacji, warstwę danych i interfejs użytkownika. Wykorzystanie Entity Framework Core umożliwia efektywne zarządzanie danymi w bazie danych, a ASP.NET Core Identity zapewnia bezpieczeństwo i zarządzanie użytkownikami. Projekt charakteryzuje się modularnością, łatwością rozbudowy i elastycznością, co pozwala na łatwe dodawanie nowych funkcji, jak np. system rekomendacji książek czy integracje z zewnętrznymi systemami. Dobrze zorganizowana struktura aplikacji sprawia, że jest łatwa do utrzymania, a potencjalne rozszerzenia są łatwe do wdrożenia. System jest solidnym fundamentem do dalszego rozwoju, spełniającym wymagania współczesnych systemów bibliotecznych.