Studia i Materiały Informatyki Stosowanej, Tom 18, Nr 1, 2026
str. 19-24

APLIKACJA DO ZARZĄDZANIA SYSTEMEM BIBLIOTECZNYM Z INTEGRACJĄ RFID

Patrycja Wołowicz, Patryk Marchewka, Piotr Kotlarz

Uniwersytet Kazimierza Wielkiego w Bydgoszczy
Instytut Informatyki
ul. Kopernika 1, 85-064 Bydgoszcz
E-mail: patrycja.wolowicz@student.ukw.edu.pl

Streszczenie. *W artykule przedstawiono projekt i implementację systemu zarządzania biblioteką, składającego się z aplikacji desktopowej dla pracowników biblioteki oraz aplikacji mobilnej dla czytelników. System opracowano z wykorzystaniem technologii C#/WPF oraz Java/Android, ze wspólną bazą danych Microsoft SQL Server. Zaimplementowano mechanizm uwierzytelniania z użyciem technologii RFID oraz bezpieczne przechowywanie haseł z wykorzystaniem algorytmu SHA256 z losową solą. Przeprowadzono analizę istniejących rozwiązań bibliotecznych (Koha, Aleph, BiblioteQ, Evergreen) i zaproponowano alternatywę charakteryzującą się niższym progiem wejścia oraz intuicyjnym interfejsem użytkownika. System umożliwia automatyzację procesów bibliotecznych, w tym monitorowanie terminów zwrotów, naliczanie kar oraz obsługę listy oczekujących na niedostępne pozycje.*

Słowa kluczowe: *system biblioteczny, RFID, C#, WPF, Java, Android, Entity Framework, MSSQL, aplikacja mobilna*

Abstract. *This paper presents the design and implementation of a library management system consisting of a desktop application for library staff and a mobile application for readers. The system was developed using C#/WPF and Java/Android technologies, with a shared Microsoft SQL Server database. An authentication mechanism using RFID technology was implemented, along with secure password storage using the SHA256 algorithm with a random salt. An analysis of existing library solutions (Koha, Aleph, BiblioteQ, Evergreen) was conducted, and an alternative characterized by a lower entry threshold and an intuitive user interface was proposed. The system enables automation of library processes, including monitoring return deadlines, calculating fines, and handling waiting lists for unavailable items.*

Keywords: *library system, RFID, C#, WPF, Java, Android, Entity Framework, MSSQL, mobile application*

1. Wstęp

Współczesne biblioteki stoją przed wyzwaniem digitalizacji procesów i automatyzacji obsługi czytelników. Rosnące oczekiwania użytkowników dotyczące dostępności usług online oraz konieczność efektywnego zarządzania zasobami wymuszają wdrażanie zintegrowanych systemów informatycznych [12].

Istniejące rozwiązania, takie jak Koha [1], Aleph [2], BiblioteQ [3] czy Evergreen [4], oferują rozbudowaną funkcjonalność, jednak ich wdrożenie wymaga znaczących nakładów finansowych oraz wsparcia dedykowanego działu IT. Małe i średnie biblioteki często nie dysponują zasobami niezbędnymi do implementacji i utrzymania tak złożonych systemów.

Celem niniejszej pracy było zaprojektowanie i zaimplementowanie systemu zarządzania biblioteką, który stanowiłby alternatywę dla złożonych rozwiązań komercyjnych. Założono utworzenie intuicyjnego interfejsu dostępnego dla użytkowników

o zróżnicowanych kompetencjach technicznych oraz integrację nowoczesnych technologii, takich jak RFID (Radio-Frequency Identification) do bezdotykowej identyfikacji użytkowników.

System składa się z dwóch komplementarnych aplikacji: desktopowej przeznaczonej dla pracowników biblioteki oraz mobilnej dedykowanej czytelnikom. Obie aplikacje komunikują się ze wspólną bazą danych, zapewniając spójność informacji.

2. Przegląd istniejących rozwiązań

Przed przystąpieniem do projektowania systemu przeprowadzono analizę porównawczą dostępnych rozwiązań do zarządzania bibliotekami. Tabela 1 przedstawia zestawienie kluczowych cech analizowanych systemów.

Tabela 1. Porównanie systemów bibliotecznych

System	Licencja	Aplik. mob.	Złożoność
Koha	Open Source	Tak	Wysoka
Aleph	Komercyjna	Tak	Bardzo wysoka
BiblioteQ	Open Source	Nie	Średnia
Evergreen	Open Source	Częściowo	Wysoka
Proponowany	**Open Source**	**Tak**	**Niska**

Koha jest najpopularniejszym otwartoźródłowym systemem bibliotecznym, oferującym pełną funkcjonalność ILS (Integrated Library System). Wymaga jednak instalacji na serwerze Linux z bazą MySQL/MariaDB oraz znajomości administracji systemowej [1].

Aleph stanowi rozwiązanie komercyjne dedykowane dużym bibliotekom akademickim. Charakteryzuje się wysokimi kosztami licencji oraz koniecznością wsparcia technicznego producenta [2].

BiblioteQ to lekka aplikacja desktopowa z interfejsem Qt, niewymagająca serwera. Jej ograniczeniem jest brak natywnej aplikacji mobilnej oraz ograniczone możliwości pracy wielostanowiskowej [3].

Evergreen projektowany był z myślą o konsorcjach bibliotecznych. Jego architektura oparta na PostgreSQL i serwerze OpenSRF charakteryzuje się wysoką skalowalnością, lecz również znaczną złożonością wdrożenia [4].

Na podstawie przeprowadzonej analizy zidentyfikowano lukę rynkową – brak prostego w obsłudze systemu łączącego aplikację desktopową z mobilną, niewymagającego zaawansowanej wiedzy technicznej do wdrożenia.

3. Architektura systemu

3.1. Struktura trójwarstwowa

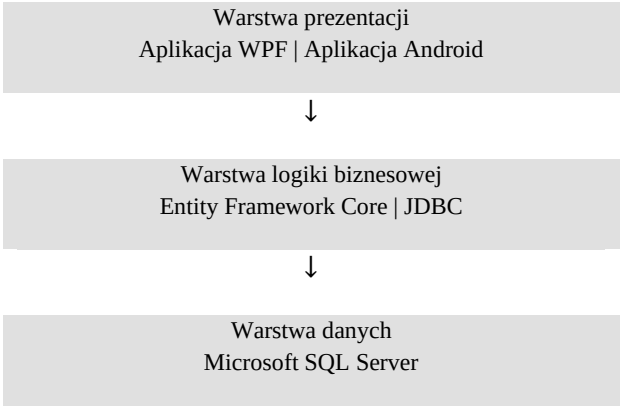
Zaprojektowano architekturę trójwarstwową, obejmującą warstwę prezentacji, logiki biznesowej oraz danych (rys. 1).

Warstwa prezentacji realizowana jest przez dwa interfejsy użytkownika:

- aplikację desktopową (C#/WPF) – dla pracowników biblioteki,
- aplikację mobilną (Java/Android) – dla czytelników.

Warstwa logiki biznesowej wykorzystuje Entity Framework Core [9] w aplikacji desktopowej oraz sterownik JDBC (jtds) w aplikacji mobilnej do komunikacji z bazą danych.

Warstwa danych oparta została na Microsoft SQL Server z wykorzystaniem języka T-SQL [7] do definiowania procedur składowanych i mechanizmów integralności.



Rys. 1. Architektura trójwarstwowa systemu

3.2. Model bazy danych

Zaprojektowano relacyjną bazę danych składającą się z 12 tabel. Tabela 2 przedstawia główne encje systemu wraz z ich przeznaczeniem.

Tabela 2. Główne encje bazy danych

Encja	Przeznaczenie
Użytkownicy	Dane kont, RFID, hash hasła, sól
Książki	Katalog pozycji, ISBN, dostępność
Autorzy	Dane autorów książek
Wypożyczenia	Historia i aktywne wypożyczenia
Recenzje	Oceny i opinie czytelników
Zaległości	Kary finansowe użytkowników
Powiadomienia	Lista oczekujących na książki
Logi	Dziennik zdarzeń systemowych
Reporty	Zgłoszenia naruszeń regulaminu

Relację wiele-do-wiele między książkami a autorami zrealizowano poprzez tabelę asocjacyjną `HashKsiążkiAutorzy` z kluczem kompozytowym. Dodatkowo zdefiniowano tabele słownikowe: `Gatunki`, `Języki`, `Role`, `Statusy` oraz `Płeć`.

Dla zapewnienia integralności danych zastosowano:

- klucze obce (FOREIGN KEY) z kaskadowym usuwaniem,
- ograniczenia unikalności (UNIQUE) dla loginu, ISBN i RFID,
- ograniczenia CHECK dla walidacji ocen (zakres 1–5),
- indeksy warunkowe dla kolumn dopuszczających NULL.

Listing 1 przedstawia utworzenie indeksu unikalnego z obsługą wartości NULL dla identyfikatora RFID.

Listing 1. Indeks unikalny z obsługą NULL

```
CREATE UNIQUE NONCLUSTERED INDEX idx_RFID_notnull
ON Uzytkownicy(RFID)
WHERE RFID IS NOT NULL;
```

4. Implementacja mechanizmów bezpieczeństwa

4.1. Hashowanie haseł z solą

W celu zabezpieczenia danych uwierzytelniających zaimplementowano mechanizm hashowania haseł z wykorzystaniem algorytmu SHA256 oraz losowej soli (ang. *salt*). Podejście to eliminuje podatność na ataki słownikowe oraz wykorzystanie tablic tęczowych (ang. *rainbow tables*).

Listing 2 przedstawia implementację hashowania hasła.

Listing 2. Hashowanie hasła z solą (C#)

```
byte[] salt = new byte[32];
using (var rng = RandomNumberGenerator.Create())
    rng.GetBytes(salt);
string salted = password +
Convert.ToBase64String(salt);
byte[] hash =
SHA256.HashData(Encoding.UTF8.GetBytes(salted));
```

Każdy użytkownik posiada unikalną sól przechowywaną w bazie danych, co uniemożliwia identyfikację identycznych haseł na podstawie porównania ich skrótów.

4.2. Uwierzytelnianie RFID

Zaimplementowano alternatywną metodę logowania z wykorzystaniem technologii RFID. Do odczytu identyfikatorów zastosowano czytnik MFRC522 podłączony do platformy Arduino, komunikujący się z aplikacją desktopową przez port szeregowy (USB).

Proces uwierzytelniania obejmuje:

1. odczyt kodu RFID z karty użytkownika,
2. transmisję kodu przez port szeregowy,
3. weryfikację identyfikatora w bazie danych,
4. utworzenie sesji użytkownika.

Listing 3 przedstawia obsługę odczytu RFID.

Listing 3. Obsługa czytnika RFID (C#)

```
void SerialPort_DataReceived(object sender,

SerialDataReceivedEventArgs e) {
    string rfidCode =
serialPort.ReadLine().Trim();
    Dispatcher.Invoke(() =>
ValidateRFIDLogin(rfidCode));
}
```

4.3. Skanowanie kodów kreskowych

Do automatycznego wprowadzania danych książek wykorzystano bibliotekę ZXing.NET [6]. Listing 4 przedstawia skanowanie ISBN.

Listing 4. Skanowanie kodu ISBN (C#)

```
var reader = new BarcodeReader();
var result = reader.Decode(CaptureFrame());
if (result != null)
    SearchBookByISBN(result.Text);
```

5. Funkcjonalność systemu

5.1. Aplikacja desktopowa

Aplikacja dla pracowników biblioteki zaimplementowana w technologii WPF [8] oferuje następujące funkcjonalności:

- zarządzanie kontami użytkowników (tworzenie, edycja, blokowanie),
- katalogowanie książek z możliwością skanowania kodów kreskowych ISBN (biblioteka ZXing.NET [6]),
- zarządzanie wypożyczeniami i zwrotami,
- rozpatrywanie zgłoszeń użytkowników,
- monitorowanie zaległości finansowych,
- generowanie raportów (eksport do CSV),
- przeglądanie dziennika zdarzeń systemowych.

Do skanowania kodów kreskowych wykorzystano bibliotekę ZXing.NET, umożliwiającą przechwytywanie obrazu z kamery i dekodowanie kodów ISBN.

5.2. Aplikacja mobilna

Aplikacja dla czytelników opracowana dla systemu Android [5] umożliwia:

- przeglądanie katalogu z wyszukiwaniem i sortowaniem,
- wypożyczanie dostępnych książek,
- zapisywanie na listę oczekujących,
- dodawanie, edycję i usuwanie recenzji,
- zgłaszanie nieodpowiednich treści,
- monitorowanie historii wypożyczeń i zaległości,
- edycję danych profilu.

Komunikacja z bazą danych realizowana jest za pomocą sterownika JDBC (jtds). Listing 5 przedstawia metodę połączenia.

Listing 5. Połączenie JDBC z bazą (Java)

```
public Connection connect() throws Exception {
    StrictMode.setThreadPolicy(new StrictMode

.ThreadPolicy.Builder().permitAll().build());

Class.forName("net.sourceforge.jtds.jdbc.Driver");
    String url = "jdbc:jtds:sqlserver://" + server
                 + ":" + port + "/" + database;
    return DriverManager.getConnection(url, user,
pass);
}
```

5.3. System logowania zdarzeń

Zaimplementowano mechanizm rejestrowania zdarzeń systemowych z trzema poziomami ważności:

- poziom 1 – operacje użytkownika (logowanie, wypożyczenia),
- poziom 10 – wychwycone błędy obsługiwane przez aplikację,
- poziom 100 – błędy krytyczne wymagające interwencji.

System logów umożliwia administratorowi analizę aktywności i diagnozowanie problemów.

6. Testy funkcjonalne

Przeprowadzono kompleksowe testy funkcjonalne obejmujące wszystkie scenariusze użycia systemu. Tabela 3 przedstawia wybrane przypadki testowe.

Tabela 3. Wybrane przypadki testowe

Scenariusz	Wynik
Logowanie z poprawnymi danymi	✓
Logowanie przez RFID	✓
Wyszukiwanie książek po tytule	✓
Wypożyczenie dostępnej książki	✓
Zapis na listę oczekujących	✓
Dodanie recenzji	✓
Naliczenie kary za przekroczenie terminu	✓
Eksport raportu do CSV	✓

Wszystkie 40 zdefiniowanych scenariuszy testowych zakończyło się wynikiem pozytywnym, potwierdzając poprawność implementacji.

7. Podsumowanie i wnioski

W ramach pracy zaprojektowano i zaimplementowano system zarządzania biblioteką składający się z aplikacji desktopowej i mobilnej. Osiągnięto następujące cele:

1. utworzono intuicyjny interfejs użytkownika dostępny dla osób o zróżnicowanych kompetencjach technicznych,
2. zautomatyzowano procesy biblioteczne, w tym monitorowanie terminów i naliczanie kar,
3. zintegrowano technologię RFID umożliwiającą szybkie uwierzytelnianie,
4. zapewniono spójność danych między aplikacjami poprzez wspólną bazę danych,
5. zaimplementowano mechanizmy bezpieczeństwa (hashowanie haseł z solą).

System stanowi alternatywę dla złożonych rozwiązań komercyjnych, oferując niższy próg wejścia oraz możliwość rozbudowy.

8. Kierunki dalszego rozwoju

Zidentyfikowano następujące obszary potencjalnego rozwoju systemu:

Zaawansowane raportowanie i analiza – planowana jest implementacja narzędzi do analizy wykorzystania biblioteki, w tym statystyk najczęściej wypożyczanych książek i najpopularniejszych autorów. Przewiduje się integrację bibliotek wizualizacyjnych (Telerik, LiveCharts) oraz eksport danych do formatów PDF i Excel.

Rozbudowa modelu ról – wprowadzenie pośrednich ról użytkowników: kustosza (dostęp do większości funkcji zarządzania bez uprawnień systemowych) oraz moderatora odpowiedzialnego za monitorowanie treści generowanych przez użytkowników (recenzje).

Inteligentne rekomendacje – implementacja chatbota jako wirtualnego asystenta zapewniającego wsparcie 24/7 oraz mechanizmów sztucznej inteligencji do generowania spersonalizowanych rekomendacji książek na podstawie historii wypożyczeń i recenzji.

Wypożyczanie książek cyfrowych – rozszerzenie systemu o dostęp do wybranych pozycji w formacie PDF z zabezpieczeniami uniemożliwiającymi pobieranie plików, kopiowanie tekstu oraz wykonywanie zrzutów ekranu. Planowany jest wbudowany czytnik z pamięcią postępu czytania.

Personalizacja profilu i treści użytkowników – możliwość zmiany zdjęcia profilowego oraz dodawania własnych podsumowań i rekomendacji książek ze słowami kluczowymi umożliwiającymi późniejsze wyszukiwanie.

Bibliografia

1. Koha Library Software, https://koha-community.org/.
2. Ex Libris Aleph Integrated Library System, https://exlibrisgroup.com/products/aleph-integrated-library-system/.

3. BiblioteQ, https://textbrowser.github.io/biblioteq/.
4. Evergreen ILS, https://evergreen-ils.org/about-us/.
5. Android Studio, https://developer.android.com/studio/intro.
6. ZXing.Net, https://github.com/micjahn/ZXing.Net/.
7. Ben-Gan I., Podstawy języka T-SQL: Microsoft SQL Server 2022, Promise, 2023.
8. Sells C., Griffiths I., Programming WPF, O'Reilly Media, 2007.
9. Lerman J., Miller R., Programming Entity Framework: DbContext, O'Reilly, 2012.
10. Skeet J., C# in Depth: Fourth Edition, Manning, 2019.
11. Schildt H., Java. Kompendium programisty. Wydanie XII, Helion, 2018.
12. Sommerville I., Software Engineering, Pearson, 2015.