POLITECHNIKA WROCŁAWSKA

WYDZIAŁ ELEKTRONIKI

**System obsługi biblioteki zintegrowany z relacyjną bazą danych wykorzystującą MySQL**

|  |  |
| --- | --- |
| **Autorzy:**  Paweł Idziak, 208766  Krzysztof Mikucki, 211530  Patryk Witkowski 218167  **E-mail kontaktowy:** 208766@student.pwr.edu.pl | **Prowadzący zajęcia:**  Dr inż. Robert Wójcik, W4/K-9 |
| **Ocena z projektu:** | |

Wrocław 2016

Spis treści

[Spis rysunków 3](#_Toc469746640)

[1. Wstęp 4](#_Toc469746641)

[1.1. Cel Projektu 4](#_Toc469746642)

[1.2. Zakres projektu 4](#_Toc469746643)

[2. Analiza wymagań 5](#_Toc469746644)

[2.1. Opis działania i schemat logiczny systemu 5](#_Toc469746645)

[2.2. Wymagania funkcjonalne 6](#_Toc469746646)

[2.2.1. Diagram przypadków użycia 7](#_Toc469746647)

[2.2.2. Scenariusze wybranych przypadków użycia 7](#_Toc469746648)

[2.3. Wymagania niefunkcjonalne 12](#_Toc469746649)

[2.3.1. Wykorzystywane technologie i narzędzia 12](#_Toc469746650)

[2.3.2. Wymagania dotyczące rozmiaru bazy danych 12](#_Toc469746651)

[2.3.3. Wymagania dotyczące bezpieczeństwa systemu 12](#_Toc469746652)

[2.4. Przyjęte założenia projektowe 13](#_Toc469746653)

[3. Projekt systemu 14](#_Toc469746654)

[3.1. Projekt bazy danych 14](#_Toc469746655)

[3.1.1. Analiza rzeczywistości i uproszczony model konceptualny 14](#_Toc469746656)

[3.1.2. Model logiczny 14](#_Toc469746657)

[3.1.3. Model fizyczny i ograniczenia integralności danych 15](#_Toc469746658)

[3.2. Projekt aplikacji użytkownika 16](#_Toc469746659)

[3.2.1. Architektura aplikacji 16](#_Toc469746660)

[3.2.2. Interfejs graficzny i struktura menu 16](#_Toc469746661)

[3.2.3. Projekt wybranych funkcji systemu 18](#_Toc469746662)

[3.2.4. Metoda podłączania do bazy danych – integracja z bazą danych 20](#_Toc469746663)

[3.2.5. Projekt zabezpieczeń na poziomie aplikacji 20](#_Toc469746664)

[4. Implementacja systemu 21](#_Toc469746665)

[4.1. Realizacja bazy danych 21](#_Toc469746666)

[4.1.1. Tworzenie tabel i definiowanie ograniczeń 21](#_Toc469746667)

[4.1.2. Implementacja mechanizmów przetwarzania danych 23](#_Toc469746668)

[4.1.3. Implementacja uprawnień 24](#_Toc469746669)

[4.2. Realizacja elementów aplikacji 24](#_Toc469746670)

[4.2.1. Obsługa menu 24](#_Toc469746671)

[4.2.2. Implementacja interfejsu dostępu do bazy danych 24](#_Toc469746672)

[4.2.3. Implementacja wybranych funkcjonalności systemu 25](#_Toc469746673)

[5. Testowanie systemu 27](#_Toc469746674)

[5.1. Instalacja i konfigurowanie systemu 27](#_Toc469746675)

[5.2. Testowanie opracowanych funkcji systemu 27](#_Toc469746676)

[5.2.1. Testowanie funkcji wyszukiwania książek 27](#_Toc469746677)

[5.2.2. Testowanie funkcji rezerwowania książek 28](#_Toc469746678)

[5.2.3. Testowanie funkcji wyszukiwania użytkowników 30](#_Toc469746679)

[5.2.4. Testowanie funkcji statystyk 31](#_Toc469746680)

[5.2.5. Testowanie funkcji sprawdzania historii wypożyczeni 31](#_Toc469746681)

[5.2.6. Testowanie dodawania książki do bazy danych 32](#_Toc469746682)

[5.2.7. Testowanie dodawania autora do bazy danych 34](#_Toc469746683)

[5.2.8. Testowanie dodawania gatunku do bazy danych 36](#_Toc469746684)

[5.2.9. Testowanie dodawania wydawnictwa do bazy danych 38](#_Toc469746685)

[5.3. Inne testy 40](#_Toc469746686)

[5.4. Wnioski z testów 40](#_Toc469746687)

[6. Podsumowanie 41](#_Toc469746688)

[Literatura 42](#_Toc469746689)

# Spis rysunków

[Rysunek 1. Schemat komunikacji 6](#_Toc469746224)

[Rysunek 2. Wymagania funkcjonalne 7](#_Toc469746225)

[Rysunek 3. Przypadki użycia 8](#_Toc469746226)

[Rysunek 4. Wymagania niefunkcjonalne 13](#_Toc469746227)

[Rysunek 5. Model konceptualny 15](#_Toc469746228)

[Rysunek 6. Model logiczny 15](#_Toc469746229)

[Rysunek 7. Model fizyczny 16](#_Toc469746230)

[Rysunek 8. Okno logowania/rejestracji. 17](#_Toc469746231)

[Rysunek 9. Okno logowania. 17](#_Toc469746232)

[Rysunek 10. Okno rejestracji. 18](#_Toc469746233)

[Rysunek 11. Okno główne - zakładka Librarian. 18](#_Toc469746234)

[Rysunek 12. Przykład wyszukania książki. 19](#_Toc469746235)

[Rysunek 13. Zamian danych przez użytkownika. 20](#_Toc469746236)

[Rysunek 14. Wyszukanie użytkownika przez administratora. 20](#_Toc469746237)

[Rysunek 15. Tabela dane 22](#_Toc469746238)

[Rysunek 16. Tabela konto 22](#_Toc469746239)

[Rysunek 17. Tabel użytkownik 23](#_Toc469746240)

[Rysunek 18. Tabela gatunek 23](#_Toc469746241)

[Rysunek 19. Tabela wydawnictwo 23](#_Toc469746242)

[Rysunek 20. Tabela autor 23](#_Toc469746243)

[Rysunek 21. Tabela rezerwacja 23](#_Toc469746244)

[Rysunek 22. Tabela książka 24](#_Toc469746245)

[Rysunek 23. Tabela wypożyczenie 24](#_Toc469746246)

[Rysunek 24. Menu główne 25](#_Toc469746247)

[Rysunek 25. Test - wyszukiwanie książek – stan bazy danych 28](#_Toc469746248)

[Rysunek 26. Test - wyszukiwanie książek - aplikacja 29](#_Toc469746249)

[Rysunek 27. Test - rezerwacja książki - stan bazy danych „przed” (książka) 29](#_Toc469746250)

[Rysunek 28. Test - rezerwacja książki - stan bazy danych „przed” (rezerwacja) 29](#_Toc469746251)

[Rysunek 29. Test - rezerwacja książki - aplikacja (dodanie książki) 30](#_Toc469746252)

[Rysunek 30. Test - rezerwacja książki - aplikacja (rezerwacja) 30](#_Toc469746253)

[Rysunek 31. Test - rezerwacja książki - stan bazy danych „po” (książka) 30](#_Toc469746254)

[Rysunek 32. Test - rezerwacja książki - stan bazy danych „po” (rezerwacja) 31](#_Toc469746255)

[Rysunek 33. Test - wyszukiwanie użytkowników - stan bazy danych 31](#_Toc469746256)

[Rysunek 34. Test - wyszukiwanie użytkowników - aplikacja 32](#_Toc469746257)

[Rysunek 35. Test - dodawanie książki - stan bazy danych „przed" 33](#_Toc469746258)

[Rysunek 36. Test - dodawanie książki - aplikacja 34](#_Toc469746259)

[Rysunek 37. Test - dodawanie książki - stan bazy danych „po" 34](#_Toc469746260)

[Rysunek 38. Test - dodawanie autora - stan bazy danych „przed" 35](#_Toc469746261)

[Rysunek 39. Test - dodawanie autora - aplikacja 36](#_Toc469746262)

[Rysunek 40. Test - dodawanie autora - stan bazy danych „po" 36](#_Toc469746263)

[Rysunek 41. Test - dodawanie gatunku - stan bazy danych „przed" 37](#_Toc469746264)

[Rysunek 42. Test - dodawanie gatunku – aplikacja 38](#_Toc469746265)

[Rysunek 43. Test - dodawanie gatunku - stan bazy danych „po" 38](#_Toc469746266)

[Rysunek 44. Test - dodawanie wydawnictwa - stan bazy danych „przed" 39](#_Toc469746267)

[Rysunek 45. Test - dodawanie wydawnictwa - aplikacja 40](#_Toc469746268)

[Rysunek 46. Test - dodawanie wydawnictwa - stan bazy danych „po" 40](#_Toc469746269)

# Wstęp

## Cel Projektu

Celem projektu było stworzenie bazy danych przeznaczonej do obsługi wypożyczalni książek oraz implementacja aplikacji umożliwiającej dostęp do niej.

## Zakres projektu

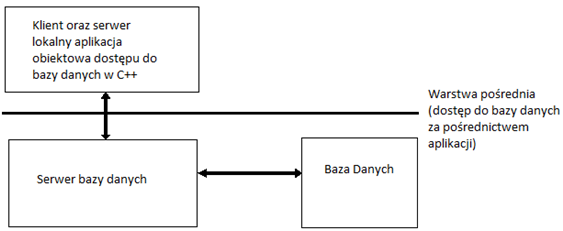
W zakres projekt wchodziło:

* Projektowanie systemu za pomocą języka UML tzn. tworzenie przypadków użycia, scenariuszy itp.
* Tworzenie bazy danych przy użyciu języka MySQL.
* Napisanie zapytań do bazy danych, widoków i procedur, które wywoływane są w aplikacji.
* Konfigurowanie lokalnej bazy danych i wypełnienie jej danymi.
* Implementacja aplikacji w języku C++ łączącej się bezpośrednio z bazą danych i pozwalającej użytkownikowi pracować na niej za pomocą interfejsu graficznego.
* Testowanie systemu.
* Pomiar czasu wykonywania procedury i widoku dla różnej ilość rekordów w bazie danych oraz analiza otrzymanych wyników.

# Analiza wymagań

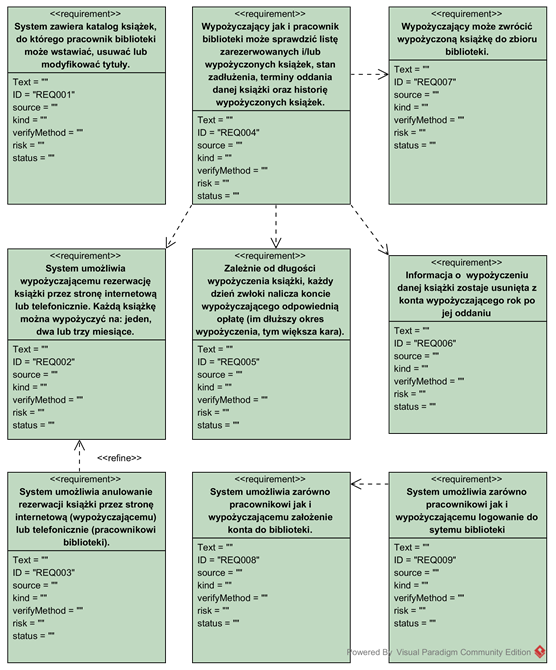
## Opis działania i schemat logiczny systemu

System umożliwiać będzie zarządzanie biblioteką w oparciu o relacyjną bazę danych (tabele opisujące dane o książkach, rezerwacjach itp.). W szczególności, dla bazy danych o znanej strukturze możliwy będzie dostęp do danych z poziomu odpowiedniej aplikacji, a także wykonywanie za pomocą tego programu określonych operacji w zależności od rodzaju użytkownika. Pracownik będzie mógł dodawać nowe książki, usuwać książki, których już nie ma w bibliotece, wyszukiwać książki oraz wypożyczać je zarejestrowanym w systemie czytelnikom. Pracownik, który jest administratorem będzie mógł dodatkowo zablokować albo usunąć konto w przypadku naruszenia przez użytkownika regulaminu. Czytelnik żeby skorzystać z oferty biblioteki będzie musiał założyć w systemie konto, za którego pośrednictwem będzie w stanie wyszukiwać książki przy pomocy rożnych kryteriów ( np. autor, tytuł, wydawnictwo itp.) oraz rezerwować książki, które nie zostały wypożyczone. Dzięki aplikacji czytelnik w dowolnym momencie będzie mógł sprawdzić termin zwrotu książek, które wypożyczył, a także datę, kiedy dostępne będą książki, które go interesowały, a które zostały wypożyczone. Każda książka może zostać wypożyczona na okres 3 miesięcy. Po tym czasie na koncie użytkownika zacznie być naliczana kara w wysokości 2 złotych za każdy miesiąc opóźnienia.



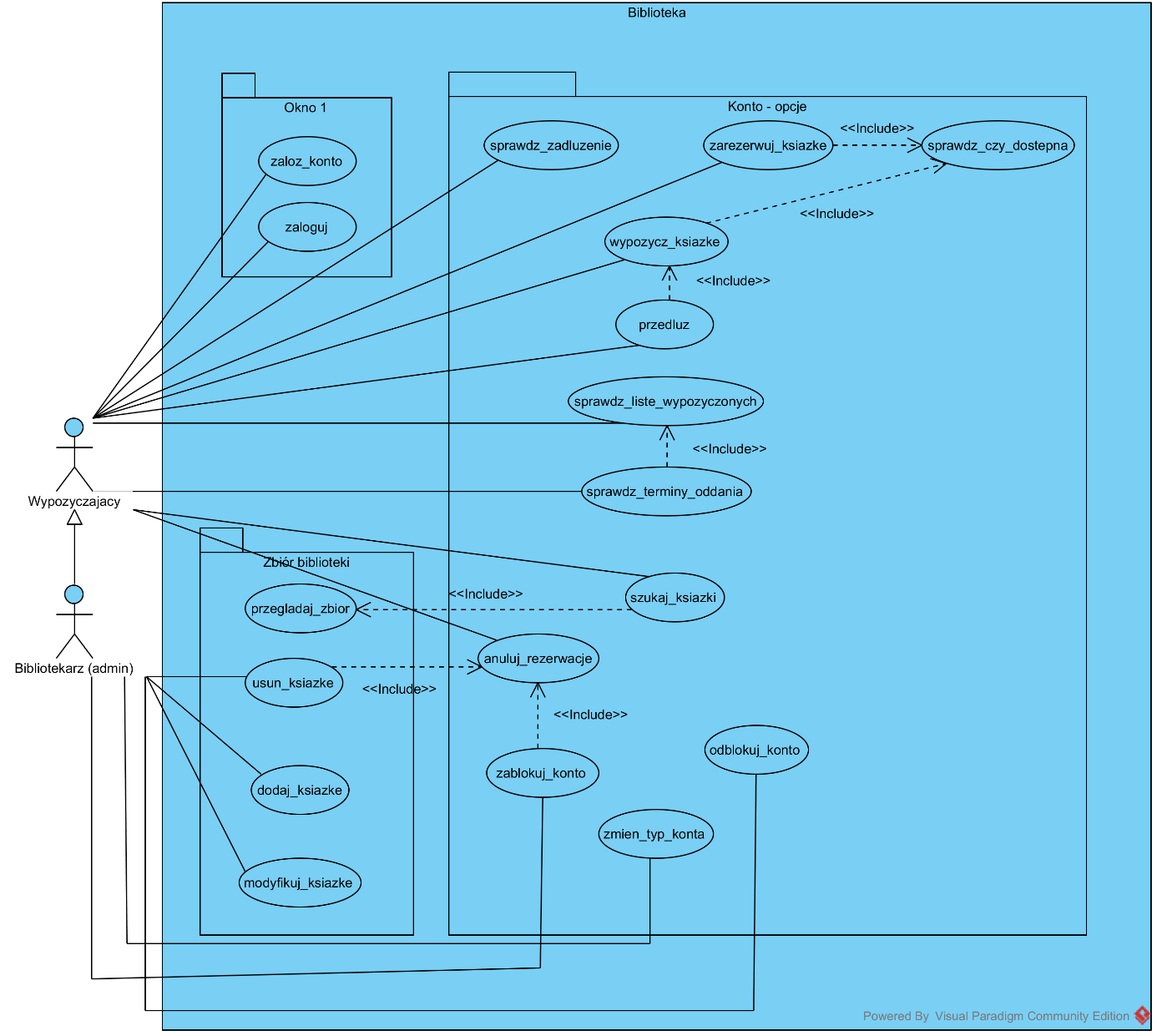
Rysunek 1. Schemat komunikacji

## Wymagania funkcjonalne



Rysunek 2. Wymagania funkcjonalne

### Diagram przypadków użycia



Rysunek 3. Przypadki użycia

### Scenariusze wybranych przypadków użycia

**Konto – tworzenie i rejestracja**

**PU Załóż konto**

CEL: Założenie nowego konta w bazie

WS (warunki wstępne): może być wywołany przez Wypożyczającego

WK (warunki końcowe): podanie loginu i hasła

PRZEBIEG:

1. Wypożyczający podaje login i hasło.

2. Jeśli istnieje już konto z podanym login, należy podać inny.

3. Jeżeli hasło jest za krótkie mnie niż (8 znaków) i nie posiada dużych i małych znaków oraz cyfr należy podać inne.

4. Do bazy zostaje dodane nowy użytkownik o podanych danych.

**PU Zaloguj**

CEL: Zalogowanie się na konto z bazy

WS (warunki wstępne): może być wywołany przez Wypożyczającego

WK (warunki końcowe): zalogowanie do systemu

PRZEBIEG:

1. Wypożyczający podaje login i hasło.

2. Jeśli konto o podanym loginie istnieje i hasło jest poprawne następuje zalogowanie na konto.

3. Jeżeli konto nie istnieje wyskakuje odpowiedni komunikat.

4. Jeżeli konto istnieje, ale hasło jest nieprawidłowe wyskakuje odpowiedni komunikat.

5. Jeżeli konto istnieje, ale zostało zablokowane wyskakuje odnowieni komunikat.

**Konto – opcje**

**PU Sprawdź zadłużenie**

CEL: Sprawdzenie zadłużenia konta

WS (warunki wstępne): może być wywołany przez Wypożyczającego

PRZEBIEG:

1. Wypożyczający sprawdza zadłużenie konta.

2. Jeśli okres wypożyczenie danej książki został przekroczony to zadłużenia konta wzrasta o 2 zł/miesiąc.

**PU Zarezerwuj książkę**

CEL: Rezerwacja książki

WS (warunki wstępne): może być wywołany przez Wypożyczającego

WK (warunki końcowe): dodanie książki do listy zarezerwowanych

PRZEBIEG:

1. Wypożyczający wybiera książkę do rezerwacji.

2. Jeśli będzie ona wypożyczona lub została już zarezerwowana w dany terminie użytkownik nie może jej zarezerwować.

3. Książka zostaje dodana do listy zarezerwowanych wraz z terminem jej wypożyczenia.

**PU Sprawdź czy dostępna**

CEL: Sprawdzenie dostępności książki w bazie danych

WS (warunki wstępne): wywoływane przez PU Zarezerwuj\_ksiazke i PU Wypozycz\_ksiazke

WK (warunki końcowe): zwraca informacje o dostępności książki

PRZEBIEG:

1. Jeżeli nie ma niewypożyczonego egzemplarzu danej książki zwracana jest odpowiedź negatywna.

2. Jeżeli jest przynajmniej jeden egzemplarz danej książki zwracana jest odpowiedź pozytywna.

**PU Wypożycz książkę**

CEL: wypożyczenie książki z bazy

WS (warunki wstępne): może być wywołany przez Wypożyczającego lub PU Przedluz

WK (warunki końcowe): oznaczenie książki w bazie danych jako wypożyczonej

PRZEBIEG:

1. Wypożyczający wybiera książkę do wypożyczenia.

2. Jeśli książka została już wypożyczona lub zarezerwowana w terminie, kiedy użytkownik chce ją wypożyczyć nie może on już tego zrobić.

3. Książka zostaje dodana do listy wypożyczonych książek przez użytkownika na okres 3 miesięcy od momentu wypożyczenia.

**PU Przedłuż**

CEL: Przedłużenie terminu wypożyczenia posiadanej książki

WS (warunki wstępne): może być wywołany przez Wypożyczającego

WK (warunki końcowe): przedłuża wypożyczenie książki na 1 miesiąc

PRZEBIEG:

1. Wypożyczający wybiera posiadaną już książkę, której termin wypożyczenia chce przedłużyć.

2. Jeżeli termin wypożyczenia już minął nie może go przedłużyć.

3. Jeżeli już raz przedłużał termin wypożyczenia danej książki to nie może tego zrobić ponownie.

4. Termin wypożyczenia zostaje przedłużony o jeden miesiąc licząc od momentu wygaśnięcia aktualnego terminu wypożyczenia.

**PU Sprawdź listę wypożyczonych**

CEL: Wyświetla listę wypożyczonych książek z bazy

WS (warunki wstępne): może być wywołany przez Wypożyczającego lub PU Sprawdz\_terminy\_oddania

WK (warunki końcowe): zwraca listę wypożyczonych książek

PRZEBIEG:

1. Wypożyczający sprawdza listę wypożyczonych przez niego książek.

**PU Sprawdź termin oddania.**

CEL: Wyświetlenie terminu oddania książki

WS (warunki wstępne): może być wywołany przez Wypożyczającego

WK (warunki końcowe): wyświetlenie odpowiedniej daty

PRZEBIEG:

1. Zostają wyświetlone terminy oddania książek w oparciu o wyniki PU Sprawdz\_liste\_wypozyczonych

**PU Szukaj książki**

CEL: Znalezienie książki o podanych kryteriach

WS (warunki wstępne): może być wywołany przez Wypożyczającego

WK (warunki końcowe): wyświetlenie książek z bazy danych

PRZEBIEG:

1. Wypożyczający podaje kryteria wyszukiwania

2. Zostaje wywołane PU Przegladaj\_zbior - zostają wyświetlone książki spełniające podane kryteria.

3. Jeżeli nie ma książki o podanych kryteria to nic nie zostanie wyświetlone.

**PU Anuluj rezerwacje**

CEL: Anulowanie istniejącej rezerwacji

WS (warunki wstępne): może być wywołany przez Wypożyczającego, PU Usuń\_książkę i PU Zablokuj\_konto

WK (warunki końcowe): usunięcie rezerwacji

PRZEBIEG:

1. Wypożyczający wybiera rezerwację

2. Rezerwacja zostaje anulowana

3. Jeżeli książka, na którą była rezerwacja została usunięta to rezerwacja też zostaje usunięta.

4. Jeżeli konto zostaje zablokowane to wszystkie rezerwacje powiązane z tym kontem zostają usunięte.

**PU Zablokuj konto**

CEL: Zablokowanie istniejącego konta

WS (warunki wstępne): może być wywołany przez Bibliotekarza

WK (warunki końcowe): zablokowanie konta

PRZEBIEG:

1. Bibliotekarz wybiera istniejące konto z bazy.

2. Jeśli nie jest zablokowane to blokuje je.

**PU Zmień typ konta**

CEL: Zmiana uprawnień dla konta

WS (warunki wstępne): może być wywołany przez Bibliotekarza

WK (warunki końcowe): zmiana uprawnień dla użytkownika

PRZEBIEG:

1. Bibliotekarz wybiera użytkownika z bazy.

2. Przydziela mu uprawnienia administratora, jeśli ich nie posiada.

3. Jeżeli użytkownik posiada uprawnienia administratora nie mogą one mu zostać odebrane.

**PU Odblokuj konto**

CEL: Odblokowanie zablokowanego konta

WS (warunki wstępne): może być wywołany przez Bibliotekarz

WK (warunki końcowe): odblokowanie konta

PRZEBIEG:

1. Bibliotekarz wybiera zablokowane konto.

2. Odblokowuje je.

**Zbiór biblioteki**

**PU Przeglądaj zbiór**

CEL: Przegląd zbioru książek z biblioteki

WS (warunki wstępne): może być wywołany przez Bibliotekarza lub PU Szukaj\_książki

WK (warunki końcowe): wyświetlenie zbioru książek

PRZEBIEG:

1. Wyświetla się zbiór wszystkich książek z bazy danych.

**PU Usuń książkę**

CEL: Usunięcie książki z bazy danych.

WS (warunki wstępne): może być wywołany przez Bibliotekarza

WK (warunki końcowe): usunięcie książki z bazy danych

PRZEBIEG:

1. Bibliotekarz wybiera książkę do uśnięcia.

2. Jeśli istnieje więcej niż jeden egzemplarz danej książki, licznik ilości egzemplarzy zostaje zmniejszony o jeden.

3. Jeżeli istnieje jeden egzemplarz danej książki zostaje ona usunięta z bazy danych.

**PU Dodaj książkę**

CEL: Dodanie nowej książki do bazy danych

WS (warunki wstępne): może być wywołany przez Bibliotekarza

WK (warunki końcowe): dodanie nowej książki do bazy danych

PRZEBIEG:

1. Bibliotekarz podaje wszystkie niezbędne dany o książce.

2. Jeżeli książka o takich danych już istnieje w bazie, zostaje zwiększony parametr odpowiadający za ilość egzemplarzy danej książki w bibliotece.

3. Jeżeli książka o podanych danych nie istnieje w bazie zostaje do niej dodana.

**PU Modyfikuj książkę**

CEL: Zmiana danych o książce

WS (warunki wstępne): może być wywołany przez Bibliotekarza

WK (warunki końcowe): zmiana wybranych informacji na temat książki

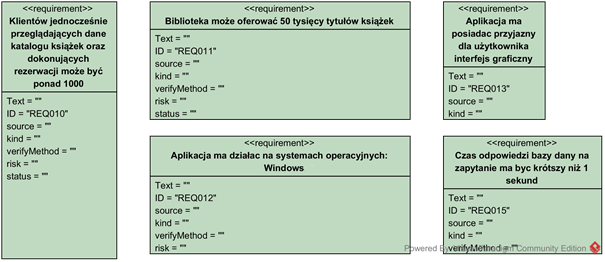
PRZEBIEG:

1. Bibliotekarz wybiera książkę do modyfikacji.

2. Bibliotekarz zmienia wybrane informacje na temat książki.

3. Zmiany muszą zostać zapisane inaczej nie zostaną one wprowadzone do bazy danych.

## Wymagania niefunkcjonalne



Rysunek 4. Wymagania niefunkcjonalne

### Wykorzystywane technologie i narzędzia

Baza danych będzie obsługiwana za pośrednictwem lokalnego serwera bazy danych MySQL (baza zostanie stworzona za pośrednictwem programu **MySQL Workbench 6.3 CE**). Interfejs użytkownika zostanie zrealizowany w postaci aplikacji obiektowej w języku **C++** uruchamianej na komputerze. Dokładniej, będzie to aplikacja z wykorzystaniem środowiska **CLR** (Common Language Runtime) udostępniona w środowisku **Visual Studio 2015** z pakietu **Visual C++**. Umożliwi ona stworzenie aplikacji okienkowej. Aby umożliwić komunikację aplikacji z bazą danych MySQL zostanie użyty „sterownik” **Connector/Net**. Jest on biblioteką dostępną w środowisku .NET. Do prezentacji funkcji systemu wykorzystany zostanie zunifikowany język modelowania **UML**.

### Wymagania dotyczące rozmiaru bazy danych

Ze względu na wymagania niefunkcjonalne baza danych będzie mogła przechowywać 50 tysięcy rekordów związanych z książkami oraz 5 tysięcy rekordów przechowujących dane na temat klientów i pracowników. Łącznie możliwy rozmiar bazy danych to 55 tysięcy rekordów.

### Wymagania dotyczące bezpieczeństwa systemu

System będzie kontrolował dostęp do bazy za pomocą metody weryfikacji użytkowników. Tylko po podaniu poprawnego hasła powiązanego z konkretnym loginem użytkownik będzie mógł skorzystać z aplikacji. Użytkownicy zarejestrowani jako administratorzy posiadać będą dodatkowe uprawnienia. Baza danych ma zaimplementowany mechanizm walidacji danych, co pozwoli uniknąć wprowadzenia nieprawidłowych informacji do tabel.

## Przyjęte założenia projektowe

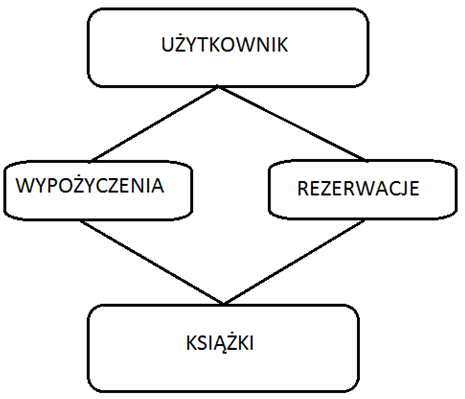
W projekcie będzie realizowany **2-warstwtowy** model komunikacji klient/serwer w postaci tzw. „cienkiego klienta”. W modelu tym przetwarzanie danych (funkcje biblioteki) jest wykonywane po stronie bazy danych (lokalnie na komputerze) za pośrednictwem zapytań SQL, natomiast po stronie klienta jest obsługiwana jedynie prezentacja danych z wykorzystaniem programu komputerowego. Dostęp do aplikacji (udostępniającej funkcje biblioteki) będzie możliwy za pośrednictwem programu komputerowego, natomiast dostęp do bazy danych będzie realizowany z wykorzystaniem funkcji programu, które komunikują się bezpośrednio z serwerem bazodanowym

# Projekt systemu

## Projekt bazy danych

### Analiza rzeczywistości i uproszczony model konceptualny

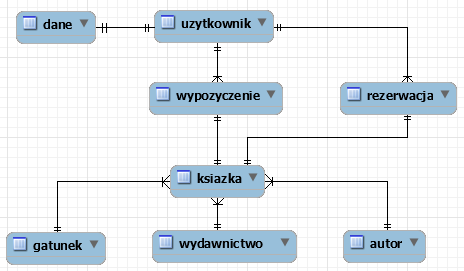
Na podstawie przedstawionej architektury powstał model konceptualny, który uwzględnia koniczność istnienia w bazie danych użytkowników, którzy muszą mieć możliwość wypożyczania lub rezerwacji (informacje o tym też muszą istnieć w bazie). Z kolei każde wypożyczenie lub rezerwacja musi być połączone z istniejąca książką.



Rysunek 5. Model konceptualny

### Model logiczny

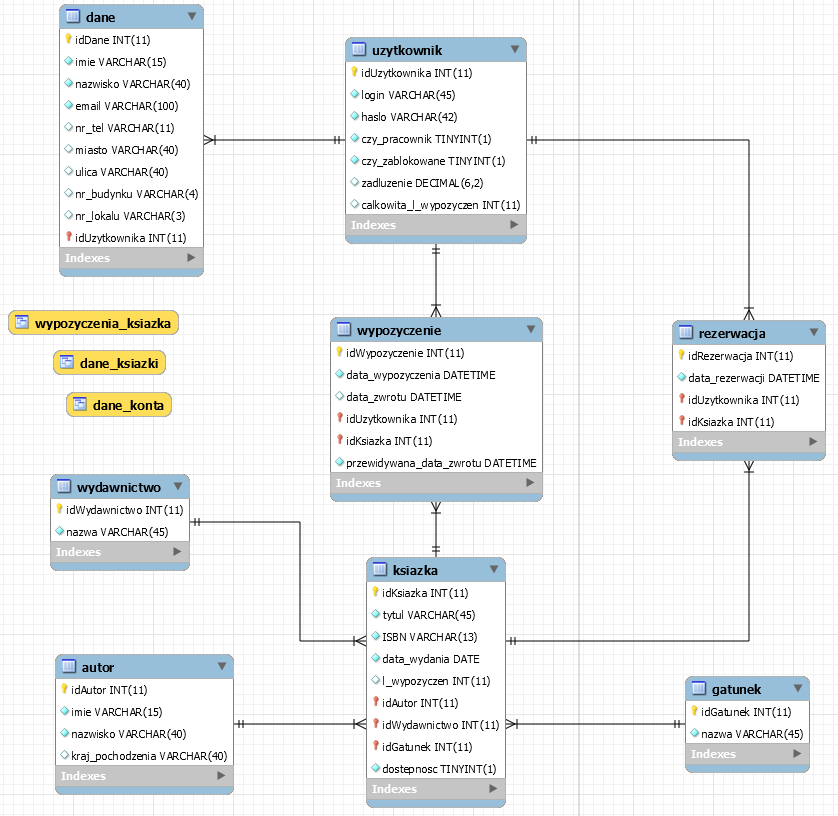
Na podstawie modelu konceptualnego powstał model logiczny uwzględniający już konkretne tabele w bazie oraz powiązania pomiędzy nimi.



Rysunek 6. Model logiczny

### Model fizyczny i ograniczenia integralności danych

Model logiczny został uzupełniony o konkretne pola w każdej tabeli wraz z typem danych – otrzymaliśmy model fizyczny.



Rysunek 7. Model fizyczny

## Projekt aplikacji użytkownika

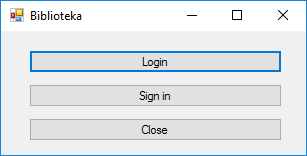
### Architektura aplikacji

Aplikacja ogranicza się do wyświetlanie interfejsu graficznego oraz wczytywaniu/zapisywaniu danych pobranych z bazy danych bezpośrednio przez zapytania SQL. Innymi słowy jest tylko graficzną warstwą pomiędzy bazą danych a klientem, sama nie wykonuje żadnych pośrednich operacji.

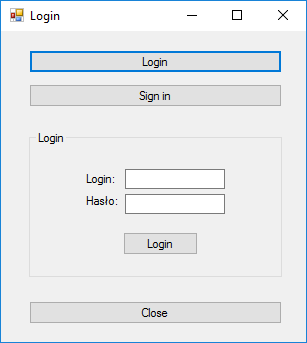
### Interfejs graficzny i struktura menu

Aplikacja składa się z dwóch głównych okien:

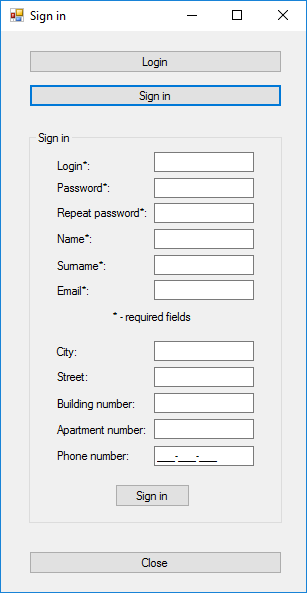
1. Okno **logowania/rejestracji** do systemu (Rysunek 8), a w nim funkcje: **logowania** (Rysunek 9), **rejestracji** (Rysunek 10) oraz **wyjścia** z programu.



Rysunek 8. Okno logowania/rejestracji.

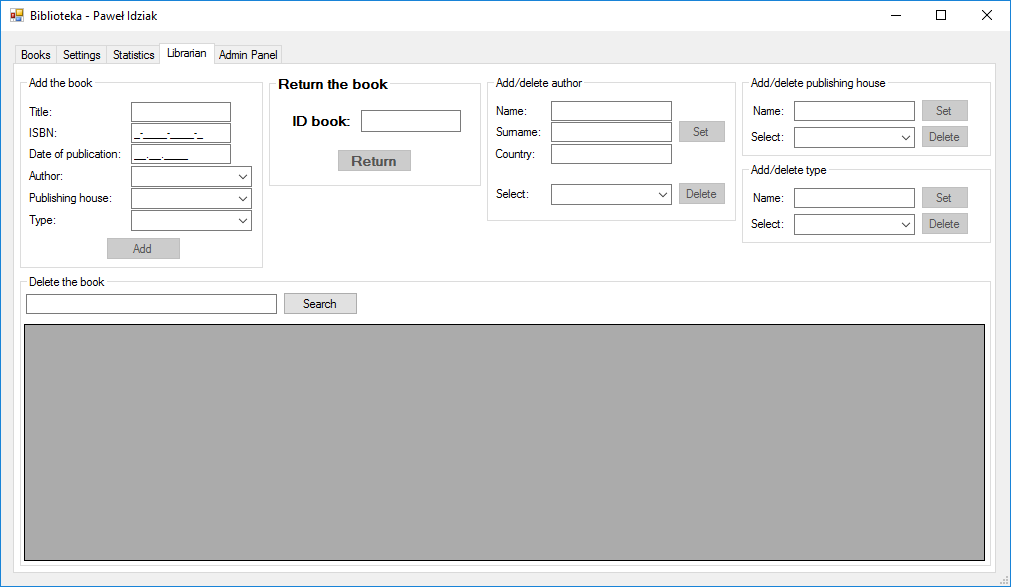


Rysunek 9. Okno logowania.



Rysunek 10. Okno rejestracji.

1. **Okno główne programu**, a w nim zakładki: **Books**, **Settings**, **Statistics**, **Librarian**, **Admin Panel**. Dla przejrzystości sprawozdania zamieszczamy jedynie wygląd zakładki *Librarian* (Rysunek 11).



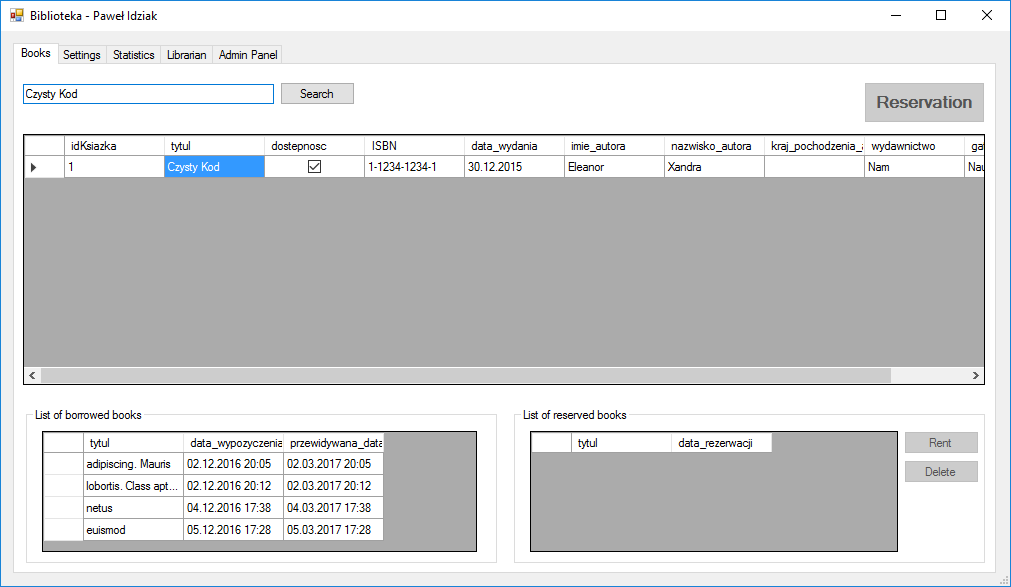
Rysunek 11. Okno główne - zakładka Librarian.

### Projekt wybranych funkcji systemu

Dla przejrzystości sprawozdania zamieszczamy jedynie kilka z funkcji systemu:

1. **Wyszukanie książki przez zalogowanego użytkownika.**

Podczas wyszukiwania książki (zakładka *Books*, przycisk *Search*) możemy podać każdy z jej atrybutów. Przykład pokazujący wyszukanie książki po tytule:



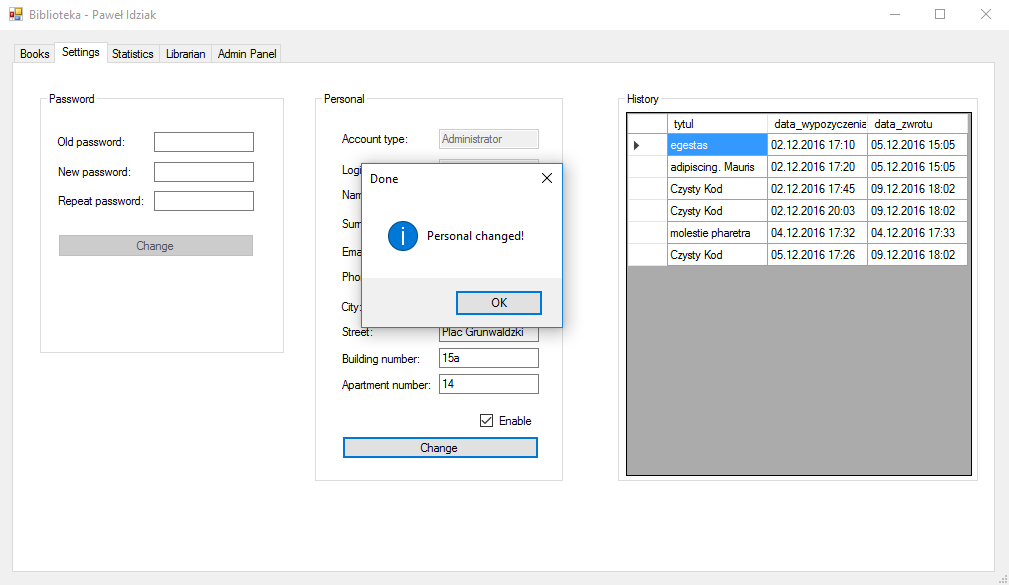
Rysunek 12. Przykład wyszukania książki.

1. **Zmiana danych personalnych przez zalogowanego użytkownika.**

Zalogowany użytkownik może zmienić swoje wszystkie dane poza typem konta oraz loginem (zakładka Settings). Zmiana hasła oraz zmiana danych personalnych odbywa się w dwóch różnych funkcjach.

**Przykład zmiany danych personalnych.**

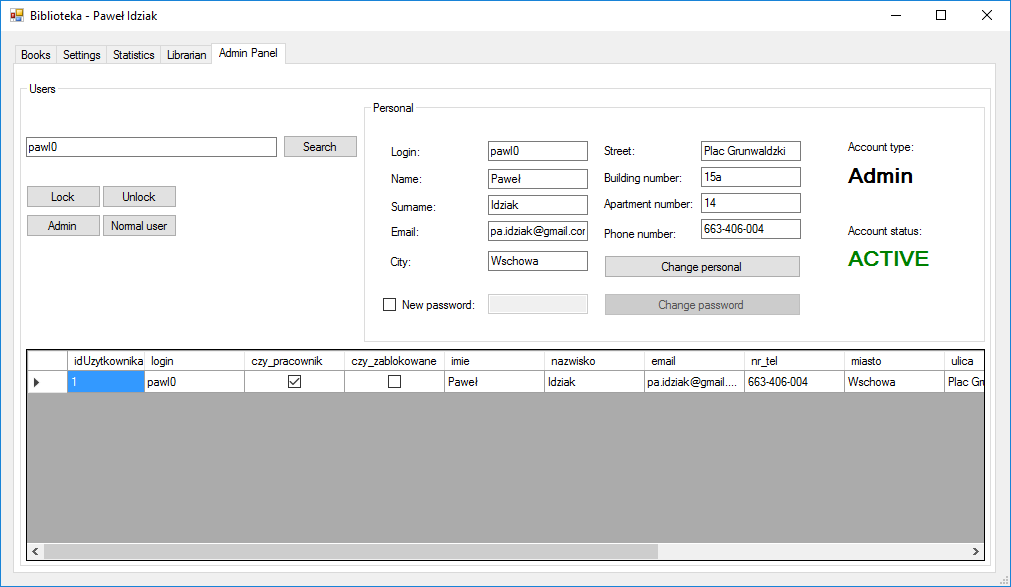
Aby zmienić swoje dane, użytkownik musi zaznaczyć pole „Enable” i wpisać odpowiednie dane do zmiany. Gdy dane będą spełniać swoje wymagania (pkt. ) przycisk „Change” zostanie odblokowany co umożliwi zatwierdzenie zmian. Po kliknięciu system dla pewności zapyta użytkownika czy na pewno chce zmienić swoje dane. Po zatwierdzeniu i poprawnym połączeniu z bazą danych system zwróci odpowiednią informację:



Rysunek 13. Zamian danych przez użytkownika.

1. **Wyszukanie użytkownika oraz zmiana jego danych przez administratora.**

Administrator ma możliwość wyszukania wszystkich istniejących użytkowników po każdym z ich atrybutów. Po wybraniu danego użytkowniku w odpowiednich polach pojawiają się jego dane, które administrator może zmienić. Dodatkowo może również zmienić typ konta wybranego użytkownika oraz je zablokować/odblokować.



Rysunek 14. Wyszukanie użytkownika przez administratora.

### Metoda podłączania do bazy danych – integracja z bazą danych

Baza danych MySQL została stworzona w programie MySQL Workbench 6.3 CE. W celach testowania i prezentacji projektu, bazę danych umieściliśmy na serwerze lokalnym. Dzięki temu możemy w prosty sposób łączyć się z bazą wprost ze środowiska Microsoft Visual Studio.

Informacje na temat dokładnego połączenia pomiędzy środowiskiem programistycznym Microsoft Visual Studio 2015, a bazą MySQL zostały przedstawione w punkcie 4.2.3.

### Projekt zabezpieczeń na poziomie aplikacji

W projekcie przyjęto następujące zabezpieczenia na poziomie aplikacji:

1. Weryfikacja podczas próby logowania do systemu – system sprawdza poprawność loginu i hasła podanego podczas logowania z danymi w bazie danych.
2. Weryfikacja poprawności pól podczas rejestracji – system sprawdza czy login, hasło etc. są odpowiedniej długości (więcej o wymaganiach dotyczących pól w pkt. ).
3. Weryfikacja poprawności pól podczas próby zmiany danych personalnych oraz hasła zarówno dla użytkownika (zakładka Settings) jak i administratora (zakładka Admin Panel). Więcej o wymaganiach w pkt. ).
4. Weryfikacja poprawności pól podczas próby dodania nowej książki, zwrócenia książki do zbioru, dodania nowego autora, gatunku lub wydawnictwa (więcej o wymaganiach w pkt. ).

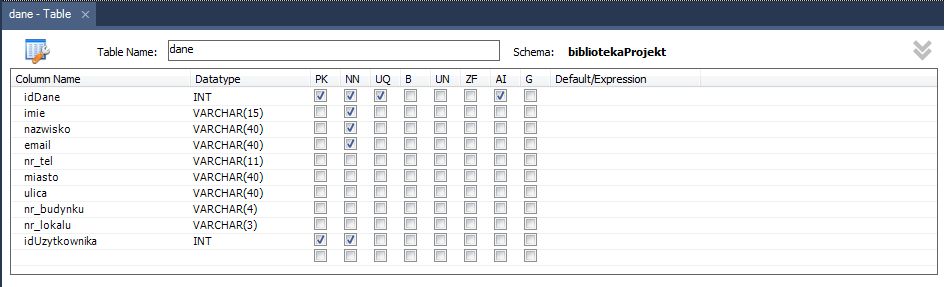
# Implementacja systemu

## Realizacja bazy danych

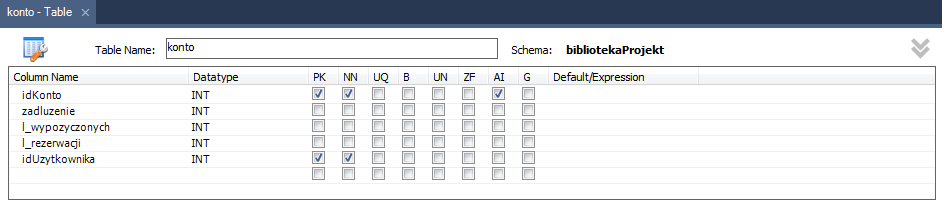
### Tworzenie tabel i definiowanie ograniczeń

Tworzenie tabel i definiowanie ograniczeń odbyło się za pomocą programu MySQL. Poniżej został przedstawiony sposób tworzenia tabel. Ograniczenia możliwe do ustawienia w MySQL to:

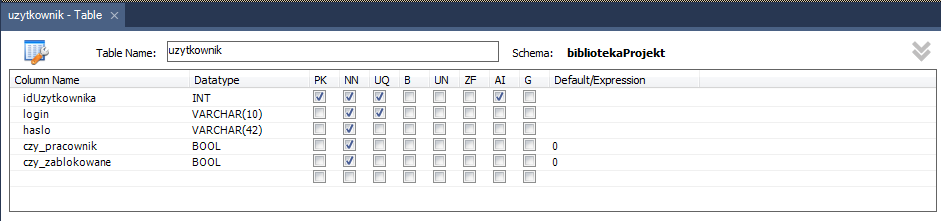
* PK – określa czy atrybut należy do klucza głównego
* NN – wartość nie może wynosić *null*
* UQ – unikalny indeks
* B – czy kolumna binarna
* UN – typ danych bez znaku
* ZF – wypełnia wartości dla danej kolumny zerami, jeśli dane dla niej są numeryczne
* AI – automatyczna inkrementacja
* G – kolumna generowana
* Default – wartość domyślna



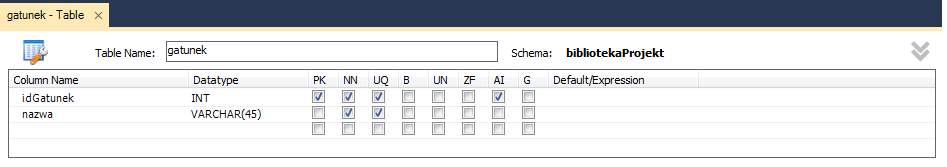
Rysunek 15. Tabela dane



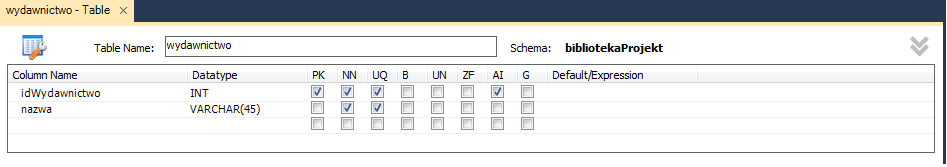
Rysunek 16. Tabela konto



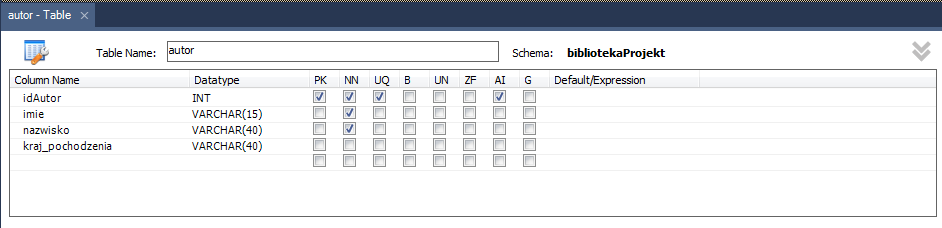
Rysunek 17. Tabel użytkownik



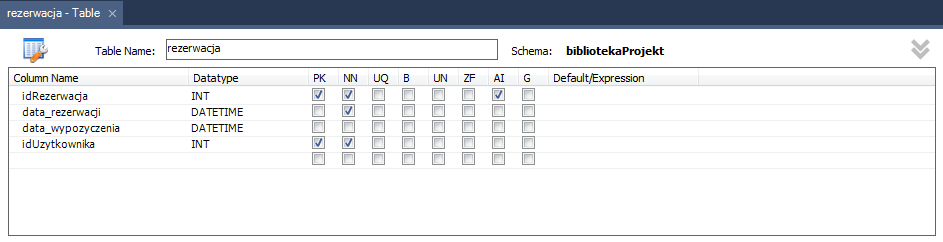
Rysunek 18. Tabela gatunek



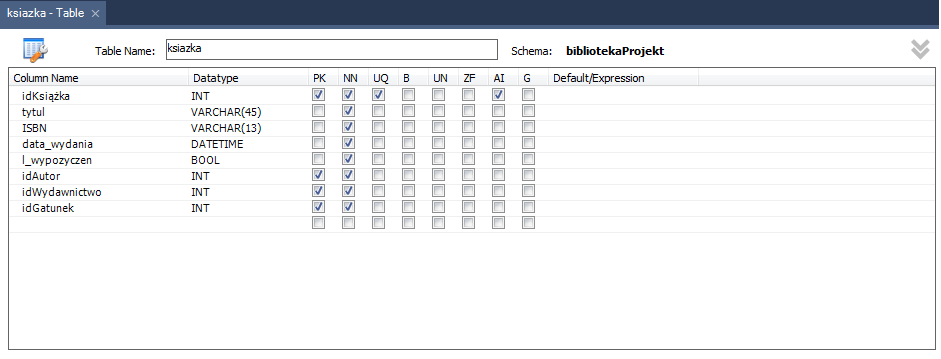
Rysunek 19. Tabela wydawnictwo



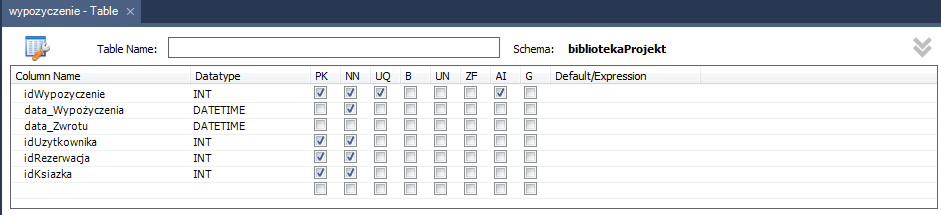
Rysunek 20. Tabela autor



Rysunek 21. Tabela rezerwacja



Rysunek 22. Tabela książka



Rysunek 23. Tabela wypożyczenie

### Implementacja mechanizmów przetwarzania danych

Do przetwarzania danych zaimplementowaliśmy procedury, a dla lepszego wczytywania – widoki. Poniżej po jednym przykładzie procedury i widoku (spis wszystkich procedur i widoków będzie wraz z innymi zapytaniami załączony w osobnym pliku).

Widok z informacjami o książkach:

CREATE VIEW 'dane\_ksiazki' AS

SELECT idKsiazka,tytul,dostepnosc,ISBN,data\_wydania,

imie AS imie\_autora, nazwisko AS nazwisko\_autora,

kraj\_pochodzenia AS kraj\_pochodzenia\_autora,

wydawnictwo.nazwa AS wydawnictwo,gatunek.nazwa AS gatunek

FROM ksiazka join gatunek on ksiazka.idGatunek = gatunek.idGatunek join

wydawnictwo on ksiazka.idWydawnictwo = wydawnictwo.idWydawnictwo join autor

on autor.idAutor = ksiazka.idAutor;

Sprawdzenie najaktywniejszego użytkownika (rosnąco):

CREATE PROCEDURE uzytkownicy\_stat()

SELECT dane.imie, dane.nazwisko, uzytkownik.login, count(idWypozyczenie) as Ilosc\_wypozyczen

FROM wypozyczenie inner join uzytkownik on uzytkownik.idUzytkownika = wypozyczenie.idUzytkownika left join dane on dane.idUzytkownika = uzytkownik.idUzytkownika

WHERE uzytkownik.idUzytkownika = wypozyczenie.idUzytkownika

GROUP BY login

ORDER BY Ilosc\_wypozyczen ASC;

### Implementacja uprawnień

W bazie rozróżniamy, w celach bezpieczeństwa, dwa rodzaje użytkowników: administrator i klient. Pierwszy z nich posiada większe uprawnienia. Aplikacja umożliwia zmianę uprawnień konta przez administratora. Poniżej odpowiadające za to zapytanie:

Zapytanie MySQL nadające prawa administratora:

UPDATE uzytkownik SET czy\_pracownik = 1 WHERE idUzytkownika = "+ id\_rekordu + ";

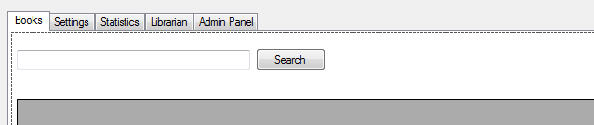
Zapytanie MySQL nadające praca zwykłego użytkownika:

UPDATE uzytkownik SET czy\_pracownik = 0 WHERE idUzytkownika = "+ id\_rekordu + ";

## Realizacja elementów aplikacji

### Obsługa menu

Obsługa menu odbywa się za pomocą zakładek.



Rysunek 24. Menu główne

* Books – wypożyczania/rezerwacja książek
* Settings – ustawienia konta
* Statistics – statystyki konta, np. ilość wypożyczonych książek itd.
* Librarian – zarządzanie książkami dla bibliotekarza
* Admin Panel – panel administratora

### Implementacja interfejsu dostępu do bazy danych

Aby mieć możliwość połączenia z bazą danych z poziomu aplikacji musimy odpowiednio zmodyfikować środowisko Microsoft Visual Studio 2015:

1. Należy zainstalować „sterownik” **Conncector/Net**

(<http://dev.mysql.com/downloads/file/?id=463758>)

1. Następnie w VS dodać odwołanie: **MySql.Data** (References -> PPM -> Add Reference..)

Po tych czynnościach mamy możliwość korzystania z wszystkich opcji dostępnych w MySql.Data. Podczas połączenia zarówno w bazie danych jak i w pliku konfiguracyjnym połączenia należy ustawić odpowiednie parametry. W naszym przypadku:

**datasource**=127.0.0.2; **port**=3306; **username**=root; **password**=root123; **database**=biblioteka

Za łączenie z bazą odpowiada poniższa linijka kodu:

MySqlConnection^ laczBaze = gcnew MySqlConnection(config);

Gdzie *config* – string z danymi konfiguracyjnymi podanymi wyżej.

### Implementacja wybranych funkcjonalności systemu

Dla przejrzystości sprawozdania przedstawiamy jedynie kilka z funkcjonalności systemu.

1. **Logowanie**

Zapytanie MySQL:

SELECT idUzytkownika, czy\_pracownik, czy\_zablokowane FROM uzytkownik

WHERE login = '" + tbLUser->Text + "'

AND haslo = PASSWORD('" + tbLPassword->Text + "');

1. **Rejestracja**

Zapytanie MySQL:

BEGIN;

INSERT INTO użytkownik (login, haslo)

VALUES('"+ tbSLogin->Text +"', PASSWORD('" + tbSPassword1->Text + "'));

INSERT INTO dane (imie, nazwisko, email, nr\_tel, miasto, ulica, nr\_budynku,

nr\_lokalu, idUzytkownika)

VALUES('"+tbSName->Text +"', '"+tbSSurname->Text +"', '"+tbSEmail->Text+ "', '"+ tbSPNumber->Text +"', '"+ tbSCity->Text + "', '" + tbSStreet->Text + "', '" + tbSBNumber->Text + "', '" + tbSANumber->Text + "', LAST\_INSERT\_ID());

COMMIT;

1. **Zmiana danych personalnych przez zalogowanego użytkownika.**

Zapytanie MySQL:

UPDATE dane SET imie = '"+ tbSName->Text +"', nazwisko = '"+ tbSSurname->Text+ "',

miasto = '"+ tbSCity->Text +"', ulica = '"+ tbSStreet->Text +"',

nr\_budynku = '"+ tbSBNumber->Text +"', nr\_lokalu = '"+ tbSANumber->Text +"',

nr\_tel = '"+ tbSPNumber->Text +"', email = '"+ tbSEmail->Text +"'

WHERE idUzytkownika = '"+ id\_uzytkownika +"'; "

1. **Wyszukiwanie książki**

Zapytanie MySQL:

CREATE VIEW 'dane\_ksiazki' AS

SELECT idKsiazka,tytul,dostepnosc,ISBN,data\_wydania,

imie AS imie\_autora, nazwisko AS nazwisko\_autora,

kraj\_pochodzenia AS kraj\_pochodzenia\_autora,

wydawnictwo.nazwa AS wydawnictwo,gatunek.nazwa AS gatunek

FROM ksiazka join gatunek on ksiazka.idGatunek = gatunek.idGatunek join

wydawnictwo on ksiazka.idWydawnictwo = wydawnictwo.idWydawnictwo join autor

on autor.idAutor = ksiazka.idAutor;

SELECT \* FROM dane\_ksiazki

WHERE CONCAT\_WS(tytul, ' ',ISBN, ' ',data\_wydania,' ',imie\_autora,' ',

nazwisko\_autora, ' ', kraj\_pochodzenia\_autora, ' ', wydawnictwo, ' ',

gatunek) LIKE '%" + tb->Text + "%' ORDER BY idKsiazka;

DROP VIEW dane\_ksiazki;

1. **Rezerwacja książki**

Zapytanie MySQL:

BEGIN;

INSERT INTO rezerwacja(data\_rezerwacji, idUzytkownika, idKsiazka)

VALUES(STR\_TO\_DATE('" + thisDay + "', '%d.%m.%Y %H:%i:%s'), " + id\_uzytkownika + ", " + id\_ksiazki + ");

UPDATE ksiazka SET dostepnosc = 0 WHERE idKsiazka = " + id\_ksiazki + ";

COMMIT;

1. **Wypożyczenie książki**

Zapytanie MySQL:

BEGIN;

INSERT INTO wypozyczenie(data\_wypozyczenia, przewidywana\_data\_zwrotu, idUzytkownika, idKsiazka)

VALUES(STR\_TO\_DATE('" + thisDay1 + "', '%d.%m.%Y %H:%i:%s'), STR\_TO\_DATE('" + dateOfReturn + "', '%d.%m.%Y %H:%i:%s'), " + id\_uzytkownika + ", " + id\_ksiazki + ");

DELETE FROM rezerwacja WHERE idKsiazka = " + id\_ksiazki + ";

UPDATE ksiazka SET l\_wypozyczen = l\_wypozyczen+1

WHERE idKsiazka = " + id\_ksiazki + ";

UPDATE uzytkownik SET calkowita\_l\_wypozyczen = calkowita\_l\_wypozyczen+1

WHERE idUzytkownika = " + id\_uzytkownika + ";

COMMIT;

1. **Wyszukiwanie użytkownika**

CREATE VIEW 'dane\_konta' AS

SELECT uzytkownik.idUzytkownika, login, czy\_pracownik, czy\_zablokowane, imie,

nazwisko, email, nr\_tel, miasto, ulica, nr\_budynku, nr\_lokalu, zadluzenie,

calkowita\_l\_wypozyczen

FROM uzytkownik join dane on uzytkownik.idUzytkownika = dane.idUzytkownika;

CREATE PROCEDURE wyszukaj\_uzytkownika(in tekst VARCHAR(255))

SELECT \* from dane\_konta

WHERE login like concat('%', tekst, '%') or

imie like concat('%', tekst, '%') or

nazwisko like concat('%', tekst, '%') or

email like concat('%', tekst, '%') or

miasto like concat('%', tekst, '%');

CALL wyszukaj\_uzytkownika('" + tbAPSearch->Text + "');

DROP VIEW dane\_konta;

DROP PROCEDURE wyszukaj\_uzytkownika;

**Wszystkie zapytania MySQL dołączamy do sprawozdania w oddzielnym wydruku.**

# Testowanie systemu

## Instalacja i konfigurowanie systemu

Na stworzenie serwera lokalnego pozwala MySQL Community Server, który należy pobrać z oficjalnej strony MySQL oraz zainstalować na docelowym komputerze (<https://dev.mysql.com/downloads/mysql/>, w razie problemów można skorzystać z MySQL Installer: <https://dev.mysql.com/downloads/installer/>).

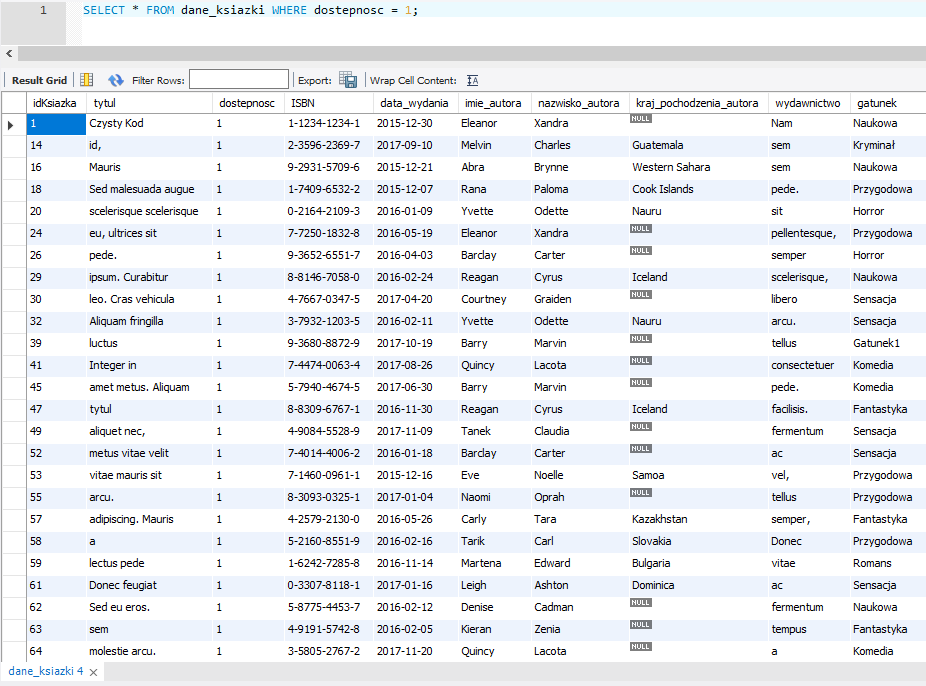
Następnie należy uruchomić projekt bazy danych i załadować ją na serwer lokalny.

Na końcu należy uruchomić aplikację, która automatycznie połączy się z serwerem lokalnym, jeżeli jego instalacja i uruchomienie przebiegły pomyślnie.

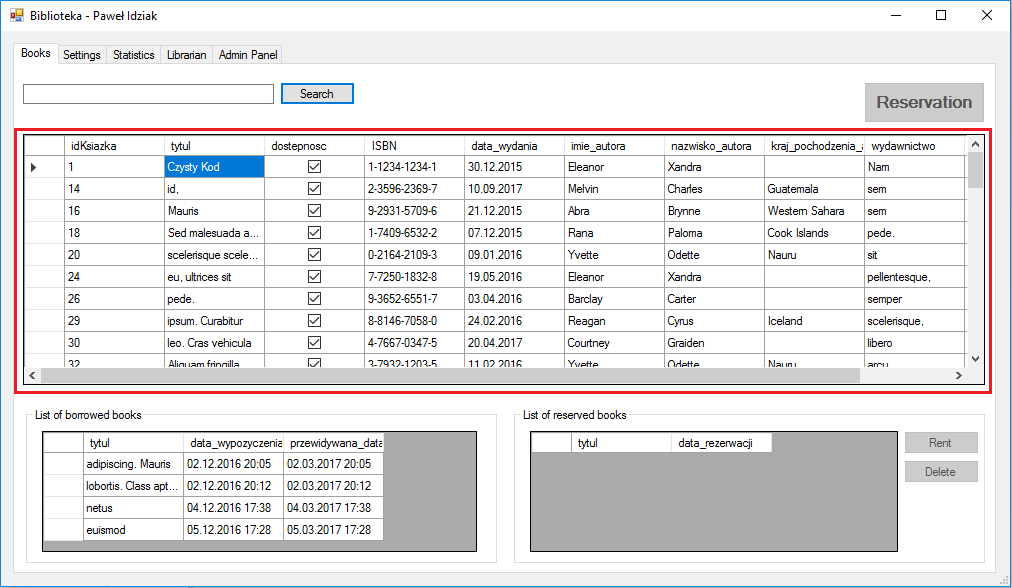
## Testowanie opracowanych funkcji systemu

### Testowanie funkcji wyszukiwania książek

Funkcja dostępna jest dla wszystkich użytkowników i działa w zaplanowany sposób. Po wpisaniu w przeznaczonym polu frazy wyszukuje wśród wszystkich książek które zawierają szukaną frazę (w tytule, numerze ISBN, imieniu bądź nazwisku autora, kraju pochodzenia autora, nazwie wydawnictwa lub w nazwie gatunku). Po nie podaniu żadnej frazy aplikacja wyświetla wszystkie dostępne aktualnie pozycje. Po podaniu nieprawidłowej frazy aplikacja wyrzuca ostrzeżenie o braku rezultatów.



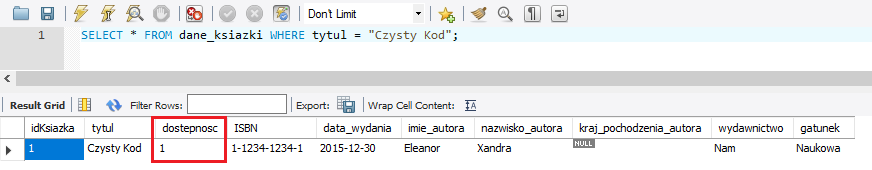
Rysunek 25. Test - wyszukiwanie książek – stan bazy danych



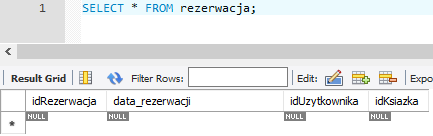
Rysunek 26. Test - wyszukiwanie książek - aplikacja

### Testowanie funkcji rezerwowania książek

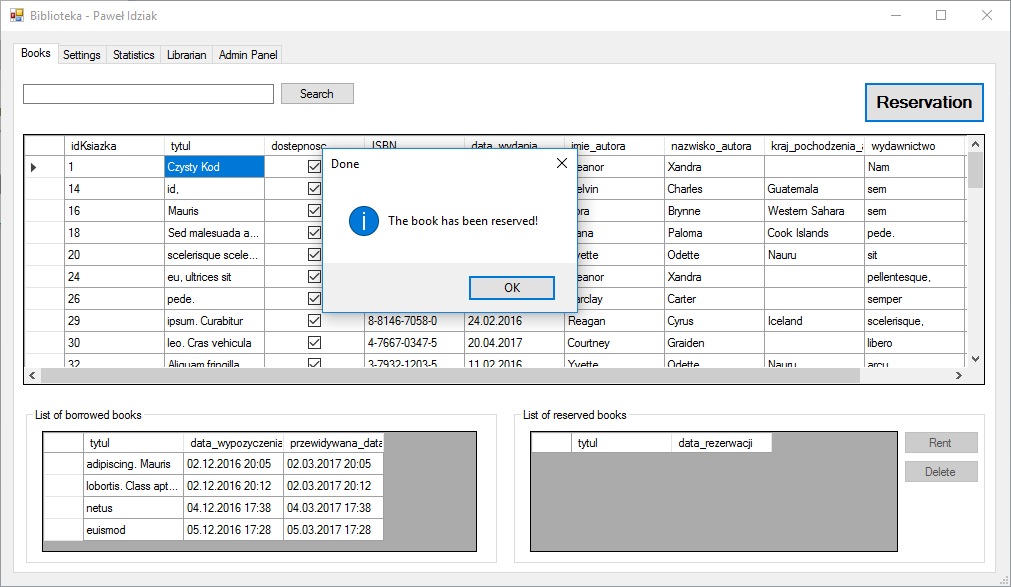
Po wyszukaniu książki można zaznaczyć interesującą nas pozycję i wcisnąć przycisk *Reservation* co powoduje, że stan książki zmienia się na niedostępny (do zarezerwowania przez innego użytkownika) i pojawia się w liście zarezerwowanych przez nas książek.



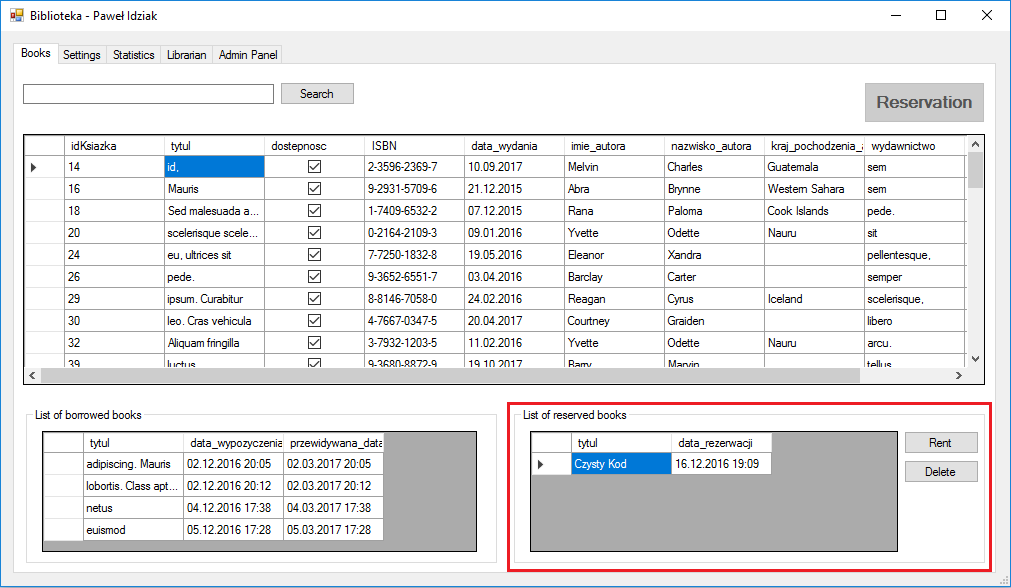
Rysunek 27. Test - rezerwacja książki - stan bazy danych „przed” (książka)



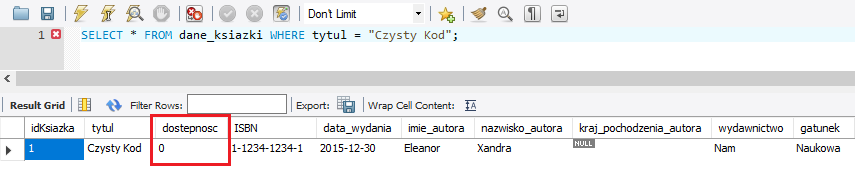
Rysunek 28. Test - rezerwacja książki - stan bazy danych „przed” (rezerwacja)



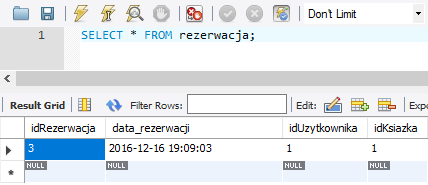
Rysunek 29. Test - rezerwacja książki - aplikacja (dodanie książki)



Rysunek 30. Test - rezerwacja książki - aplikacja (rezerwacja)



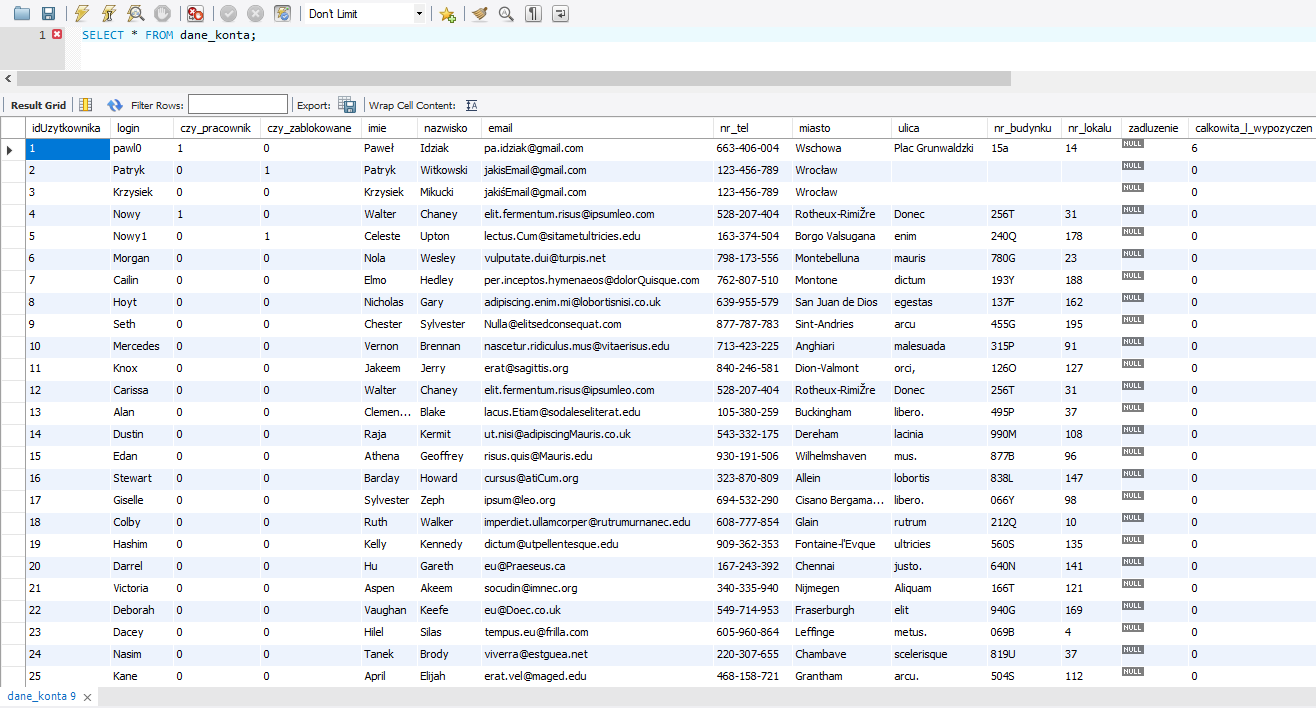
Rysunek 31. Test - rezerwacja książki - stan bazy danych „po” (książka)



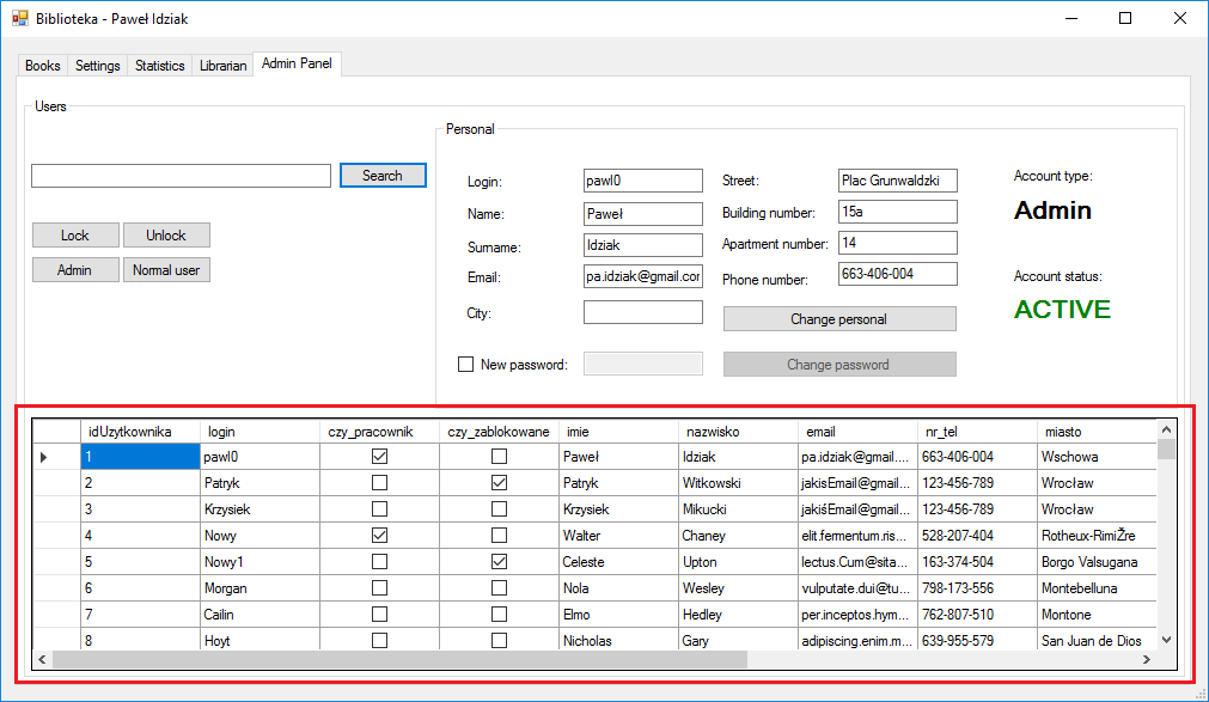
Rysunek 32. Test - rezerwacja książki - stan bazy danych „po” (rezerwacja)

### Testowanie funkcji wyszukiwania użytkowników

Funkcja dostępna jest tylko w panelu admina i działa podobnie do funkcji wyszukiwania książek. Amin może podać dowolną frazę i wyszukani zostaną wszyscy użytkownicy w których danych (login, imię, nazwisko, email, miasto, numer telefonu) zawierają wpisaną frazę. Po podaniu frazy która nie występuje u żadnego z użytkowników program wyświetla powiadomienie o braku rezultatów. Nie podanie żadnej frazy zwraca listę wszystkich użytkowników.



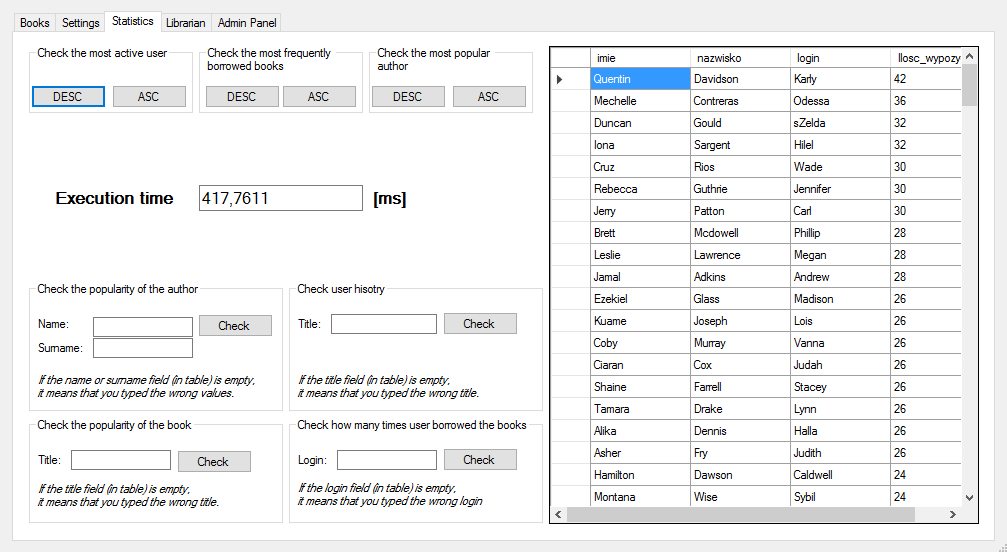
Rysunek 33. Test - wyszukiwanie użytkowników - stan bazy danych



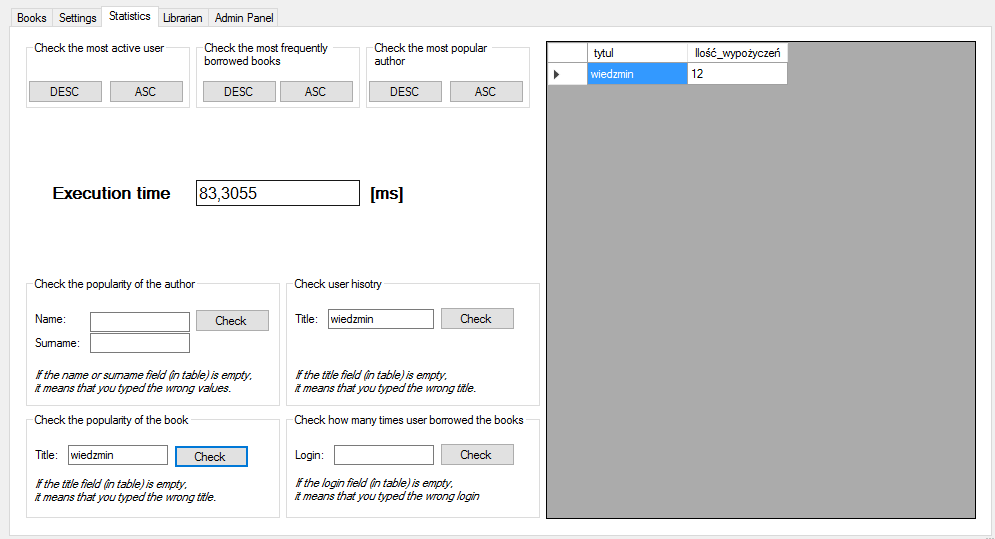
Rysunek 34. Test - wyszukiwanie użytkowników - aplikacja

### Testowanie funkcji statystyk z polami do wypełnienia

W zakładce statystyki znajdują się funkcje odpowiedzialne za wyświetlanie statystyk. Podzielone są one na dwie grupy, w jednej z nich znajdują się funkcje oparte na procedurach które wyświetlają w porządku malejącym lub rosnącym popularność danego autora, użytkownika lub tytułu (wyświetlana jest lista np. wszystkich użytkowników z informacją ile książek wypożyczyli) . Funkcje z drugiej grupy pozwalają nam na sprawdzanie popularności autora, książki lub użytkownika, wszystkie działają w bardzo podobny sposób, tj. użytkownik wpisuje w odpowiednim polu imię oraz nazwisko autora, tytuł książki lub login użytkownika, a program zwraca z bazy informacje: ile razy książki danego autora były wypożyczane, ile razy dana książka była wypożyczana lub ile wypożyczeń odnotował użytkownik. Po podaniu nieprawidłowych danych program nie wyrzuca komunikatu lecz zwraca pusty rekord z bazy, tak samo dzieje się gdy nie zostanie podana żadna fraza.



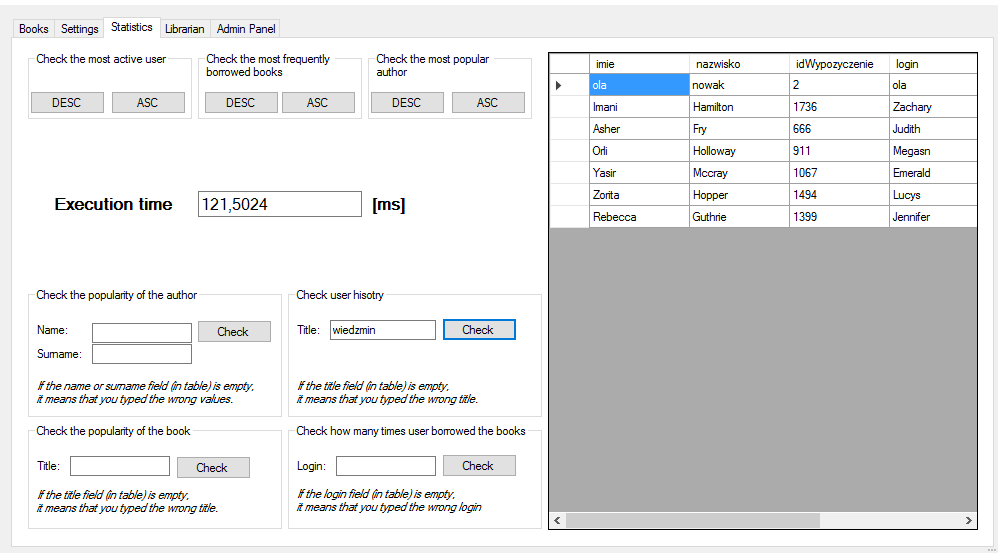
Rysunek 35 Test-statystyka wyświetlania po najaktywniejszych użytkownikach-aplikacja



Rysunek 36. Test-statystyka ilość wypożyczeń danego tytułu-aplikacja

### Testowanie funkcji sprawdzania historii wypożyczeń

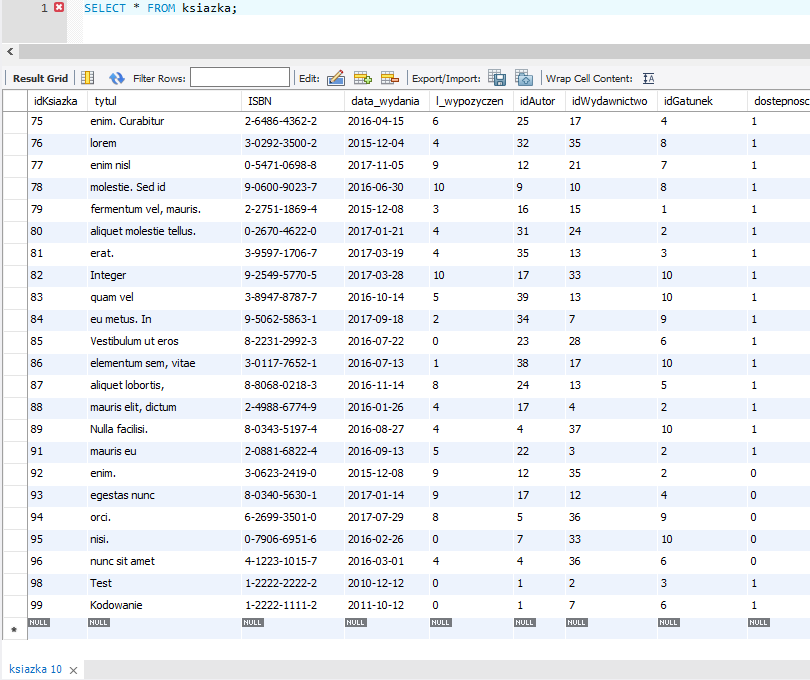
Po podaniu tytułu książki wyświetlone zostają wszystkie zanotowane w bazie wypożyczenie danego tytułu, zwracany jest login użytkownika, który wypożyczał książkę jego imię, nazwisko oraz daty wypożyczenia oraz zwrotu. Rekordy wyświetlane są według daty wypożyczenia (od najstarszej). Nie podanie żadnej wartości spowoduje wyświetlenie wszystkich zarejestrowanych wypożyczeń, natomiast podanie złej frazy spowoduje nie zwrócenie żadnego rekordu.



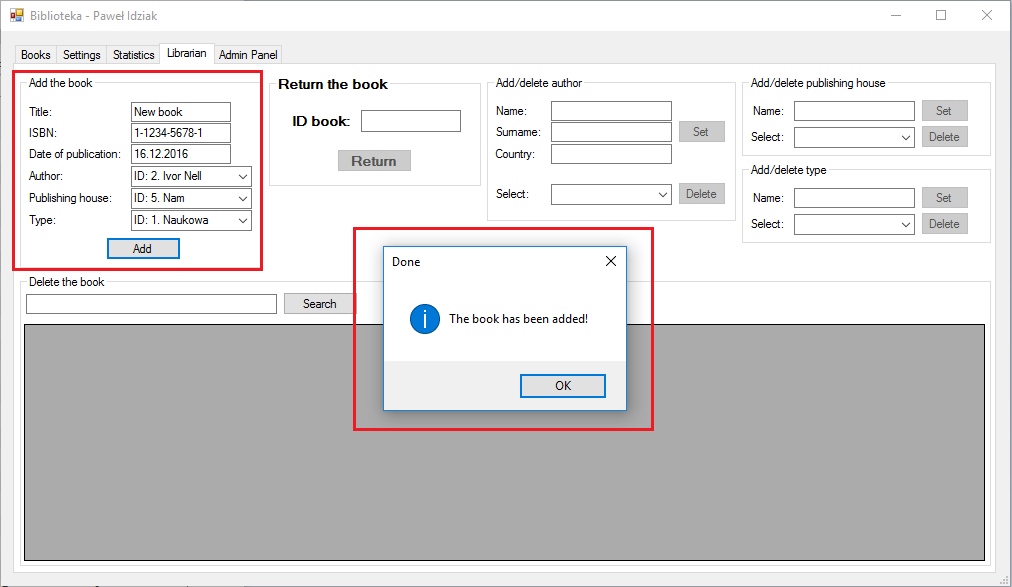
Rysunek 37. Test wyświetlania historii wypożyczeń danego tytułu

### Testowanie dodawania książki do bazy danych

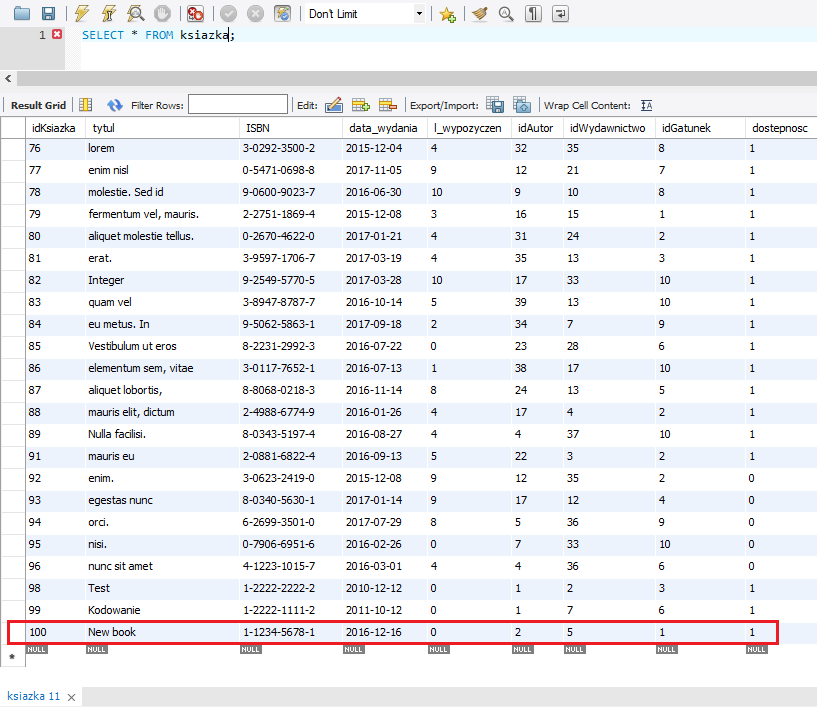
Aby do bazy danych została dodana nowa pozycja Należy w zakładce „Libarian” wypełnić odpowiednie pola, zawierające dane na temat tytułu numeru ISBN autora, wydawnictwa gatunku (w trzech ostatnich polach musza znaleźć się takie wartości jakie znajdują się już w bazie to znaczy, że nie można dodać książki danego autora jeżeli nie istnieje on w bazie jako autor). Zabezpieczenie przed wprowadzeniem niepoprawnych danych polega na odblokowaniu przycisku „Add” (odpowiedzialnego za wysłanie zapytania do bazy) dopiero po wprowadzeniu danych które zostaną zaakceptowane przez bazę.



Rysunek 38. Test - dodawanie książki - stan bazy danych „przed"



Rysunek 39. Test - dodawanie książki - aplikacja



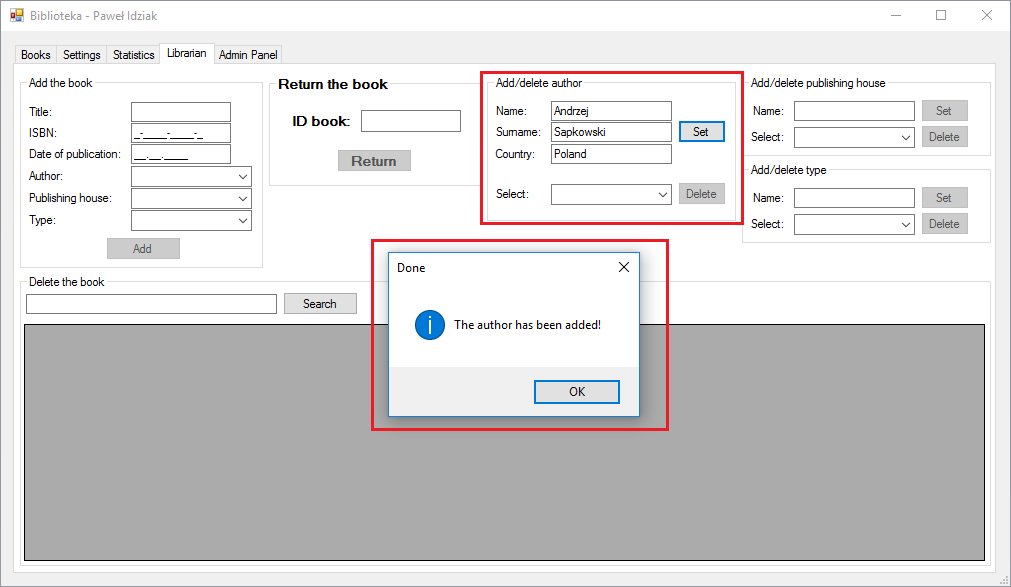
Rysunek 40. Test - dodawanie książki - stan bazy danych „po"

### Testowanie dodawania autora do bazy danych

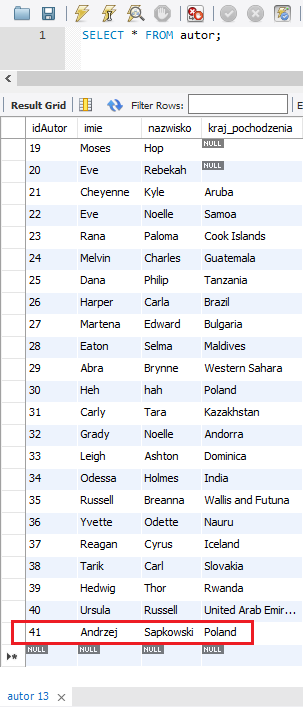
Aby dodać do bazy nowego autora do bazy danych należy w zakładce „Libarian” wypełnić odpowiednie pola w których znaleźć się musi imię oraz nazwisko autora oraz kraj jego pochodzenia. Przycisk „Set” (odpowiedzialny za wysłanie zapytania do bazy danych) dostępny jest dopiero po porwanym wypełnieniu wszystkich wymaganych pól

****

Rysunek 41. Test - dodawanie autora - stan bazy danych „przed"



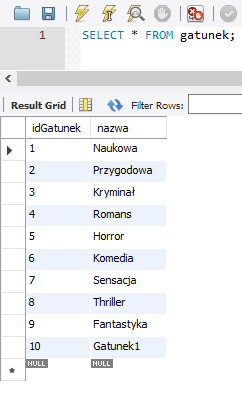
Rysunek 42. Test - dodawanie autora - aplikacja

****

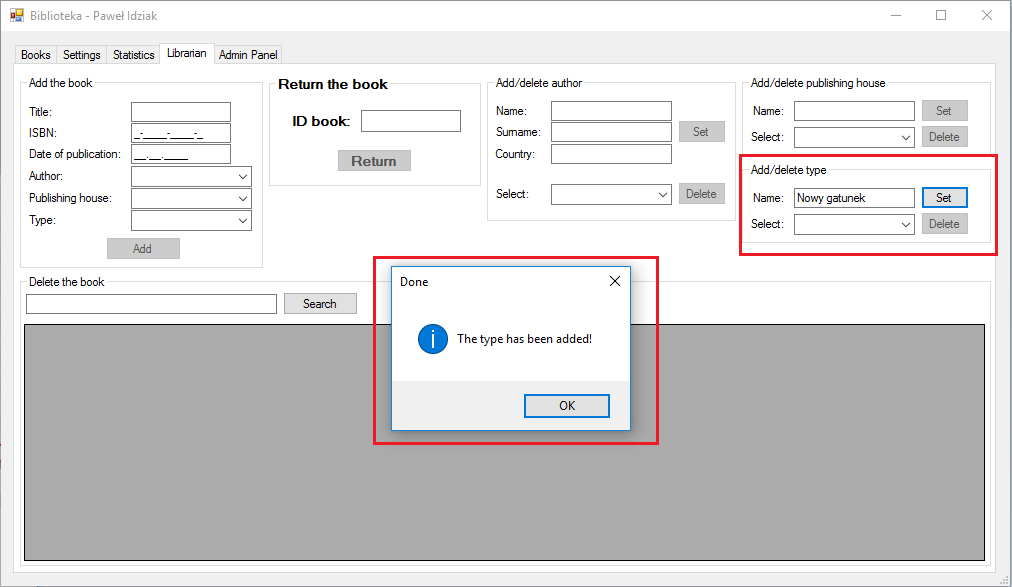
Rysunek 43. Test - dodawanie autora - stan bazy danych „po"

### Testowanie dodawania gatunku do bazy danych

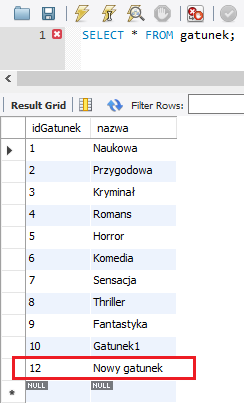
Aby dodać do bazy nowy gatunek do bazy danych należy w zakładce Libarian wypełnić odpowiednie pola w których znaleźć się musi nazwa gatunku. Przycisk „Set” (odpowiedzialny za wysłanie zapytania do bazy danych) dostępny jest dopiero po porwanym wypełnieniu wymaganego pola



Rysunek 44. Test - dodawanie gatunku - stan bazy danych „przed"



Rysunek 45. Test - dodawanie gatunku – aplikacja



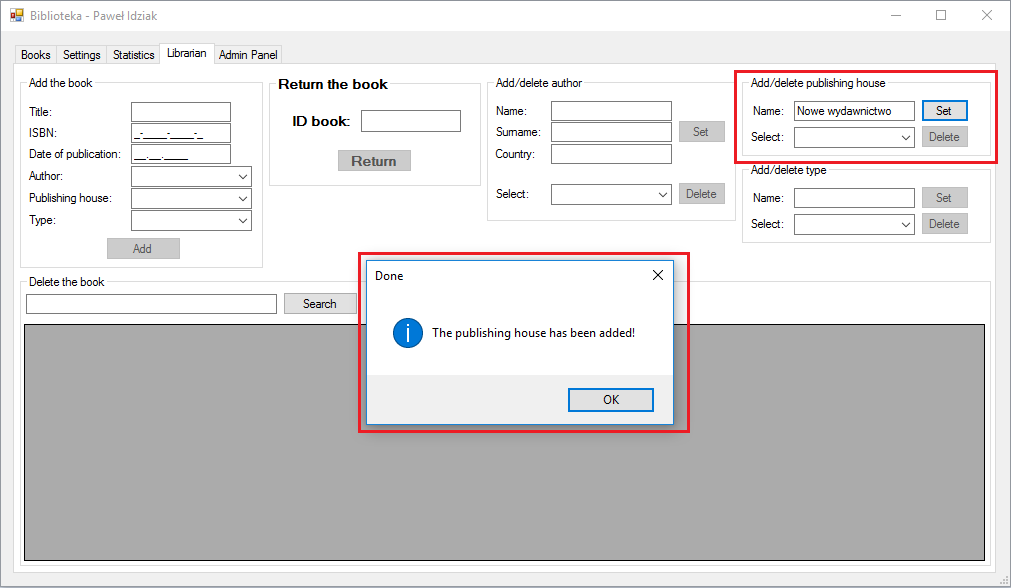
Rysunek 46. Test - dodawanie gatunku - stan bazy danych „po"

### Testowanie dodawania wydawnictwa do bazy danych

Aby dodać do bazy nowe wydawnictwo do bazy danych należy w zakładce Libarian wypełnić odpowiednie pole z nazwą wydawnictwa. Przycisk „Set” (odpowiedzialny za wysłanie zapytania do bazy danych) dostępny jest dopiero po porwanym wypełnieniu wszystkich wymaganego pola.



Rysunek 47. Test - dodawanie wydawnictwa - stan bazy danych „przed"



Rysunek 48. Test - dodawanie wydawnictwa - aplikacja



Rysunek 49. Test - dodawanie wydawnictwa - stan bazy danych „po"

## Inne testy

Na aplikacji zostały przeprowadzone testy, które miały na celu porównanie czasów wykonywania operacji takich jak SELECT, wywołanie procedur czy widoków. Zmierzone i porównane zostały czasy wyszukiwania konkretnej książki w bazie danych. Do porównania użyto zapytania SELECT z połączonymi tabelami, PROCEDURY działającej na tabelach bazy oraz procedury, która wykonywała się na widoku tworzonym podczas wywoływania zapytania w bazie, każde z zapytań dawało te same wyniki. Otrzymane wyniki przedstawiono poniżej, dla każdej liczby danych w bazie przeprowadzono 10 pomiarów czasu, z których wyciągnięto średnią.

|  |  |  |  |
| --- | --- | --- | --- |
| **Porównanie różnych metod przeszukiwania bazy czasy [ms]** | | | |
| **liczba rekordów w bazie** | **WIDOK + PROCEDURA** | **PROCEDURA** | **SELECT** |
| 200 | 18,35337 | 7,88809 | 6,03075 |
| 500 | 18,45554 | 9,05539 | 7,08577 |
| 1000 | 19,73392 | 11,13999 | 8,09051 |
| 2000 | 19,99627 | 14,82736 | 9,50986 |

Tabela 1. Porównanie wyników wykonywania różnych zapytań w bazie

*Wykres 1. Porównanie czasów wykonywania różnych zapytań do bazy*

## Wnioski z testów

Po przetestowaniu funkcji zawartych w programie stwierdzić można, że działają one w zaplanowany na początku sposób. Porównanie czasów wykonywania poszczególnych operacji na bazie wykazuje jednoznacznie, że najszybszym sposobem wyszukiwania w bazie było zapytanie select.

# Podsumowanie

Celem projektu było stworzenie bazy danych służącej do obsługi wypożyczalni książek, a także implementacja aplikacji dającej możliwość korzystania z bazy danych. Praca nad projektem została podzielona na kilka etapów. Pierwszym z nich było stworzenie architektury bazy danych za pomocą języka UML. Następnym krokiem była kreacja bazy danych przy pomocy języka MySQL oraz napisanie wszystkich niezbędnych zapytań, widoków i procedur do korzystania z bazy. Kolejnymi etapami były konfiguracja lokalnej bazy oraz implementacja aplikacji w języku C++ łączącej się bezpośrednio z bazą danych i pozwalającej użytkownikowi pracować na niej za pomocą interfejsu graficznego. Końcowym etapem było przeprowadzenie testów funkcjonalnych oraz pomiaru czasu wykonywania zapytań, wywoływania widoków i procedur.

Stworzony system umożliwia zarządzanie biblioteką w oparciu o relacyjną bazę danych (tabele opisujące dane o książkach, rezerwacjach itp.). Pracownik może dodawać nowe książki, usuwać książki, których już nie ma w bibliotece, wyszukiwać książki oraz wypożyczać je zarejestrowanym w systemie czytelnikom. Pracownik, który jest administratorem może dodatkowo zablokować albo usunąć konto w przypadku naruszenia przez użytkownika regulaminu. Czytelnik żeby skorzystać z oferty biblioteki musi założyć w systemie konto, za którego jest w stanie wyszukiwać książki przy pomocy rożnych kryteriów ( np. autor, tytuł, wydawnictwo itp.) oraz rezerwować książki, które nie zostały wypożyczone. Dzięki aplikacji czytelnik w dowolnym momencie może sprawdzić termin zwrotu książek, które wypożyczył, a także datę, kiedy dostępne będą książki, które go interesowały, a które zostały wypożyczone.

Baza danych jest obsługiwana za pośrednictwem lokalnego serwera bazy danych MySQL (została stworzona za pośrednictwem programu MySQL Workbench 6.3 CE). Interfejs użytkownika został zrealizowany w postaci aplikacji obiektowej w języku C++ uruchamianej na komputerze.

Zrealizowaliśmy 2-warstwtowy model komunikacji klient/serwer w postaci tzw. „cienkiego klienta”. W modelu tym przetwarzanie danych (funkcje biblioteki) jest wykonywane po stronie bazy danych (lokalnie na komputerze) za pośrednictwem zapytań SQL, natomiast po stronie klienta jest obsługiwana jedynie prezentacja danych z wykorzystaniem programu komputerowego.

Na końcu wykonaliśmy testy funkcjonalne, czyli sprawdzające poprawność działania podstawowych funkcji oferowanych przez system (m. in. wyszukiwanie książek, rezerwowanie książek) oraz dokonaliśmy pomiaru czasu wykonywania procedury i widoku dla różnej ilość rekordów w bazie danych. Z wykresu zależności czasu od ilości rekordów w bazie wynika, że dla rosnącej ilości informacji rośnie czas wykonywania zapytania. Wzrost ten jest jednak powolny, z czego wynika, że ilość rekordów nie ma znacznego wpływu na wykonywanie zapytań. Porównując czas wykonywania procedur i widoku widzimy, że wywoływanie procedur z widokami zajmuje o wiele więcej czasu. Spowodowane to jest w głównej mierze konstrukcją widoku. Jeżeli łączy on wiele tabel i dodatkowo wykonuje jakieś operacje, np. liczenie różnicy dni, to czas, jak pokazują to testy znacznie się wydłuża, niż w przypadku, gdy wywołamy samą procedurę lub użyjemy komendy select.

# Literatura

[1] [Wrycza](http://helion.pl/autorzy/stanislaw-wrycza,stawry.htm) S., *Język UML w modelowaniu systemów informatycznych,* Helion, Gliwice, 2016.

[2] Stones R. Matthew N., *Bazy danych i MySQL. Od podstaw,* Helion,Gliwice, 2004.

[3] Beynon-Davies P., *Systemy baz danych*, WNT, Warszawa, 2000.

[4] Grębosz J., *Symfonia C++ tom 2,* Edition 2000, Kraków, 2006.

[5] Strona internetowa: <http://wazniak.mimuw.edu.pl/index.php?title=Bazy_danych>, 2014.

[6] Strona internetowa: [https://dev.mysql.com/doc/connector-cpp/en](https://dev.mysql.com/doc/connector-cpp/en/)/ , 2016