

Politechnika Wrocławska  
Wydział Elektroniki  
Kierunek Informatyka

# Projekt Bazy danych

## **“Biblioteka wiejska”**

Osoby:

Leszek Ostek 249439

Michał Miller 241319

Michał Rajkowski 248821

Prowadzący:

Dr inż Roman Ptak

Termin:

Wtorek, 11:15-13:00

# SPIS TREŚCI

## SPIS TREŚCI

### 1. Wstęp

#### 1.1 Cel i zakres projektu

### 2. Analiza wymagań

#### 2.1. Opis biznesowy świata rzeczywistego

##### 2.1.1 Opis zasobów ludzkich

##### 2.1.2 Przepisy i strategia firmy

##### 2.1.3 Dane techniczne

#### 2.2. Wymagania stawiane tworzonej aplikacji

##### 2.2.1 Wymagania funkcjonalne

##### 2.2.2 Wymagania нефunkcjonalne

### 3. Projekt systemu

#### 3.1. Projekt bazy danych

##### 3.1.1. Model konceptualny

##### 3.1.2. Model logiczny i fizyczny

##### 3.1.3. Mechanizmy bezpieczeństwa

##### 3.1.4. Metoda połączenia z bazą danych

##### 3.1.5. Triggery

##### 3.1.6. Indeksy

##### 3.1.7. Widoki

#### 3.2. Projekt aplikacji użytkownika

##### 3.2.1. Diagram przypadków użycia

###### 3.2.1.1 Diagram przypadków użycia dla pracownika

###### 3.2.1.2 Diagram przypadków użycia dla klienta

##### 3.2.2 Interfejs graficzny i struktura menu

###### 3.2.2.1 Przykładowe menu główne

###### 3.2.2.2 Przykładowe pole do wysyłania wiadomości

###### 3.2.2.3 Menu wyboru wiadomości

###### 3.2.2.4 Tworzenie i edycja nowej książki

### 4. Implementacja systemu bazy danych

#### 4.1. Tworzenie tabel i definiowanie ograniczeń

##### 4.1.1. Komentarz do zastosowanych rozwiązań

###### 4.1.2 Kod Tabel

###### 4.1.3. Problemy napotkane przy definiowaniu tabel

#### 4.2. Implementacja mechanizmów przetwarzania danych

##### 4.2.1. Indeksy

##### 4.2.2. Widoki

#### [4.3. Testowanie bazy danych na przykładowych danych](#)

##### [4.3.1. Test widoków na przykładowych danych](#)

##### [4.3.2. Testy wydajnościowe indeksu](#)

##### [4.3.4. Napotkane problemy przy testowaniu](#)

#### [5. Implementacja i testy aplikacji](#)

##### [5.1 Instalacja i konfigurowanie systemu](#)

###### [5.1.1 Instalacja MySQL](#)

###### [5.1.2 Konfiguracja Bazy danych](#)

##### [5.2 Instrukcja użytkownika aplikacji](#)

##### [5.3 Testowanie opracowanych funkcji systemu](#)

###### [5.3.1 Testy jednostkowe](#)

##### [5.4 Omówienie wybranych rozwiązań programistycznych](#)

###### [5.4.1 Struktura kodu i implementacja wybranych funkcjonalności systemu.](#)

###### [5.4.2 Implementacja interfejsu dostępu do bazy danych](#)

###### [5.4.3 Implementacja mechanizmów bezpieczeństwa](#)

###### [5.4.3.1 System obsługi wyjątków](#)

###### [5.4.3.2 Umieszczenie aplikacji](#)

#### [6. Podsumowanie i wnioski](#)

#### [7. Literatura](#)

# 1. Wstęp

## 1.1 Cel i zakres projektu

Celem projektu jest stworzenie systemu symulującego działanie aplikacji bibliotecznej, dzięki której użytkownik może wypożyczać, książki, a pracownik za jej pomocą może sterować działaniem wypożyczalni. Biblioteka jest umiejscowiona na wsi i ma niewielu użytkowników.

Na początku naszej pracy zredagowaliśmy opis słowny naszego projektu, aby móc na nim wzorować dalsze prace. Zdefiniowaliśmy opis zasobów ludzkich, strategię firmy, dane techniczne, aby móc uzmysłowić sobie skalę naszego projektu. Wymagania funkcjonalne oraz wymagania niefunkcjonalne zdefiniowane wcześniej, pozwoliły nam dokładnie przeanalizować co musimy uwzględnić w naszej pracy ze strony technicznej, aby projekt był przydatny i zdalny do użycia. Wiedzieliśmy również, że nasz projekt musi być zrozumiany zawczasu i aby pozwolić zespołowi efektywną pracę w przyszłości.

W następnym etapie pracy stworzyliśmy projekt systemu definiując model konceptualny, przedstawiony w postaci diagramu oraz dodatkowego opisu słownego. Dodatkowo stworzyliśmy model fizyczny i logiczny odzwierciedlające w uproszczeniu układ rzeczywisty naszego projektu. Określiliśmy mechanizmy bezpieczeństwa w naszej aplikacji, których nie ma zbyt dużo ponieważ jest to mała wiejska biblioteka. Ważnym etapem tej części pracy było zdefiniowanie metody połączenia naszego programu z bazą danych. Następnie określiliśmy działanie triggerów, indeksów oraz widoków. Stworzyliśmy kilka poglądowych obrazków prezentujących jak nasza aplikacja będzie wyglądać od strony graficznej.

Następnym etapem naszej pracy było pisanie kodów SQL, naszej bazy danych. Na początku zapisaliśmy nasze tabele, po czym zaimplementowaliśmy więcej mechanizmów przetwarzania danych takich jak indeksy oraz widoki. Następnie przeszliśmy do testowania stworzonej bazy danych na przykładowych, uzupełnionych przez nas danych. Poprawie udało się stestować widoki oraz indeksy, stosując testy wydajnościowe. Do zarządzania bazą danych używaliśmy strony PHP Admin, która pozwala na szeroki zakres operacji na MySQL.

Ostatnim oraz według nas najbardziej czasochłonnym etapem projektu było stworzenie aplikacji i jej testowanie. Ma ona łączyć się z bazą danych i umożliwiać działanie i konfigurację na danych ze stworzonej wcześniej bazy danych. Stworzyliśmy wiele testów za pomocą których sprawdziliśmy poprawne działanie naszej aplikacji. Użyliśmy testów jednostkowych oraz testów wydajnościowych. Uznaliśmy, że w tym samym punkcie należytym będzie omówić zastosowane przez nas rozwiązania programistyczne. Na

samym końcu naszej pracy zaprezentujemy obserwację i wnioski wynikające z długiej pracy nad złożonym projektem.

## 2. Analiza wymagań

### 2.1. Opis biznesowy świata rzeczywistego

#### 2.1.1 Opis zasobów ludzkich

System zawiera katalog książek w którym pracownik może dodawać, modyfikować i usuwać książki. Książkę identyfikuje: tytuł, autor, liczba egzemplarzy, link do obrazka. Pracownik może dodawać, edytować i usuwać egzemplarze książki. Egzemplarz identyfikuje id egzemplarza, id książki, stan, sygnatura. Klient może przeglądać egzemplarze. Pracownik może dodawać, edytować i usuwać klientów. Klienta identyfikuje: imię, nazwisko, data urodzenia, email, hasło. Pracownik może przeglądać dane na temat klientów. Pracownik i klient mogą przeglądać katalog książek. Klient może zarezerwować dowolny egzemplarz książki. Rezerwację identyfikuje: id egzemplarza, id klienta, data.. Pracownik może dodawać, edytować lub usuwać rezerwację. Klient może tylko usunąć swoją rezerwację. Pracownik i klient mogą przeglądać rezerwację przy czym klient widzi tylko swoje rezerwację. Pracownik może utworzyć wypożyczenie jeśli wypożyczy klientowi książkę. Wypożyczenie identyfikuje: id egzemplarza, id książki, id klienta, data. Pracownik może przeglądać, usuwać, modyfikować wypożyczenia klientów przy czym klient może przeglądać tylko swoje. Pracownik i klient mogą wysyłać sobie wiadomości. Klient może wystawić opinię o książce. Klient i Pracownik mogą przeglądać inne opinie oraz zobaczyć średnią ocenę książki. Aplikacja klienta informuje klienta jeśli zalega z oddaniem książki bądź informuje go o dostępności zarezerwowanej książki.

#### 2.1.2 Przepisy i strategia firmy

Biblioteka znajduje się na wsi oraz posiada jednego starszego pracownika dlatego aplikacje pracownika i klienta muszą być maksymalnie proste w obsłudze. Biblioteka jest dosyć mała posiada około 5000 książek gdzie pośród nich występuje średnio 2 egzemplarze każdego tytułu. W ciągu roku z biblioteki korzysta średnio 200 różnych osób. Średnio wypożyczanych i oddawanych jest 5 książek dziennie.

### 2.1.3 Dane techniczne

Aplikacja pracownika oraz klienta jest przeznaczona dla systemu Windows lub Linux. Do napisania programów zostanie użyty język Java z narzędziami Swing. Aplikacje będą korzystały z relacyjnej bazy danych oraz będą w architekturze klient serwer.

## 2.2. Wymagania stawiane tworzonej aplikacji

### 2.2.1 Wymagania funkcjonalne

- System zawiera katalog książek do którego pracownik może dodawać nowe książki. Książkę identyfikuje: tytuł, autor, link do obrazka.
- Pracownik może modyfikować książkę.
- Pracownik może usunąć książkę.
- Pracownik i klient mogą przeglądać katalog książek
- Pracownik może dodać egzemplarz książki: id\_egzemplarza, id\_książki, stan, sygnatura.
- Pracownik może modyfikować egzemplarz
- Pracownik może usunąć egzemplarz
- Pracownik może przeglądać egzemplarze danej książki
- Klient może przeglądać egzemplarze danej książki
- Pracownik może dodawać nowych klientów. Klienta identyfikuje: imię, nazwisko, data urodzenia, email, hasło.
- Pracownik może modyfikować dane klienta.
- Pracownik może usunąć klienta.
- Pracownik może przeglądać dane na temat klientów.
- Klient może zarezerwować dowolny egzemplarz. Rezerwację identyfikuje: id egzemplarza, id klienta, data.
- Pracownik może modyfikować rezerwacje.
- Pracownik może usunąć rezerwację dowolnego klienta.
- Pracownik może przeglądać rezerwacje wszystkich klientów.
- Klient może przeglądać tylko swoje rezerwację.
- Pracownik może utworzyć wypożyczenie kiedy klient wypożyczy egzemplarz książki: id egzemplarza, id książki, id klienta, data oddania.
- Pracownik może wysłać klientowi wiadomość którą identyfikuje: id wiadomości, id\_klienta, data, treść.
- Pracownik może wysłać wiadomość do wszystkich klientów na raz.
- Klient może wysłać pracownikowi wiadomość
- Klient może dodać jedną opinię o danej książce którą identyfikuje: id opinii, id\_klienta, ocena (1-5), opinia słowna.
- Klient może usunąć swoją opinie
- Klient może edytować swoją opinie.
- Klient oraz pracownik mogą przeglądać opinie o danej książce.

- Klient i pracownik mogą zobaczyć średnią ocenę danej książki.
- Aplikacja klienta informuje klienta o zaleganiu z oddaniem książki.
- Aplikacja klienta informuje o możliwości odebrania książki.

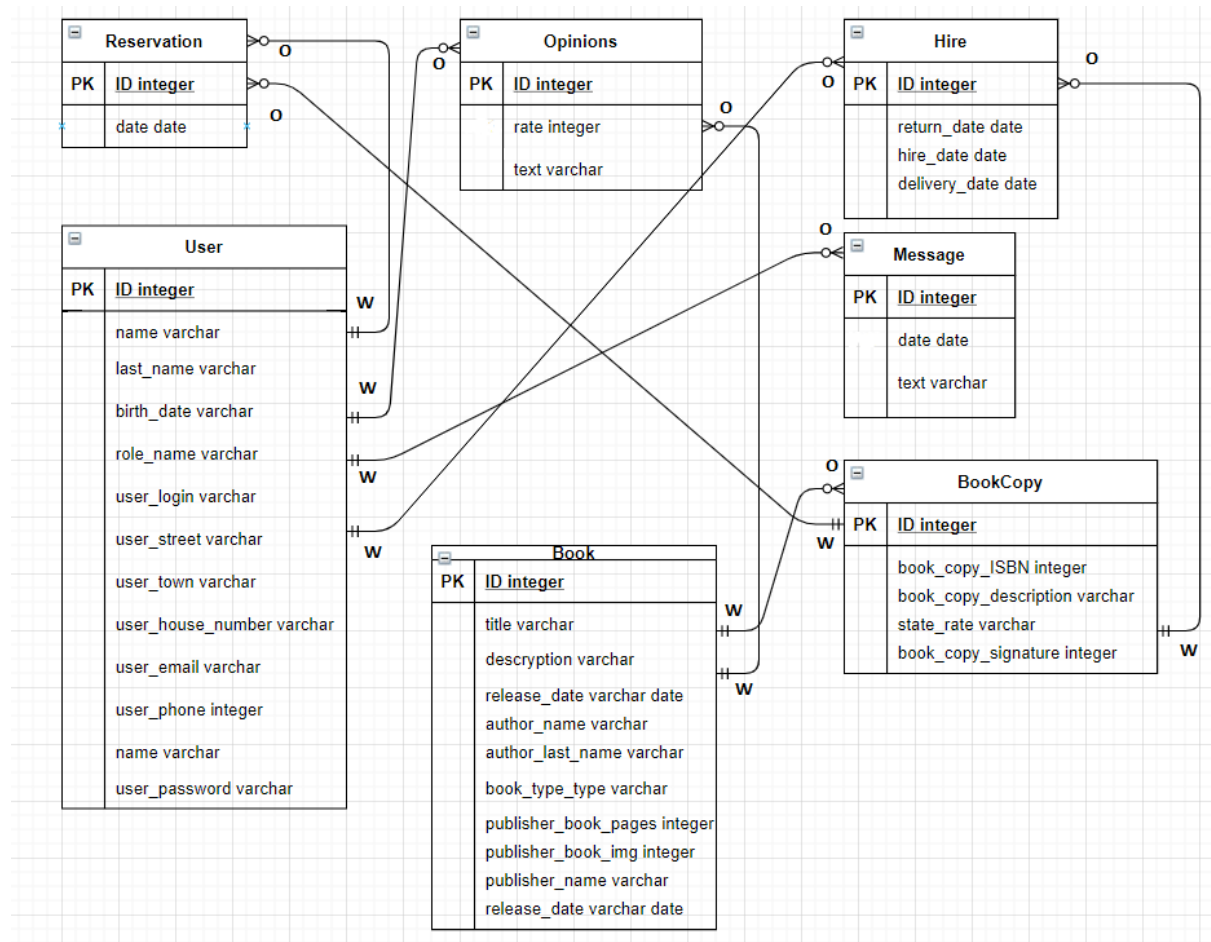
### 2.2.2 Wymagania niefunkcjonalne

- Rozróżniamy aplikację klienta oraz pracownika.
- Aplikacje działają na systemach Windows oraz Linux.
- Aplikacja ma działać w architekturze klient serwer.
- Język programowania Java
- Podejście projektowe Database first.
- Biblioteka Swing do stworzenia interfejsu graficznego
- Visual Paradigm Community Edition oraz Draw.io do tworzenia diagramów.
- W projekcie zamierzamy użyć relacyjnej bazy danych.
- Baza danych MySQL.
- Tabela książek będzie posiadała około  $5000/2 = 2500$  rekordów.
- Tabela klientów będzie posiadała około 200 rekordów po roku użytkowania w bibliotece.
- Zakładamy że bezpieczeństwo aplikacji zapewnią nam testy jednostkowe.
- Bezpieczeństwo danych zapewni nam architektura klient serwer.

## 3. Projekt systemu

### 3.1. Projekt bazy danych

#### 3.1.1. Model konceptualny



\*\*W - Uczestnictwo wymagane

\*\*O - Uczestnictwo opcjonalne

\*\*Wszystkie połączenia na diagramie to połączenia wiele do jednego

Typy danych wykorzystane w naszej aplikacji:

- **INTEGER** – typ danych związany z kluczami
- **DATE** – automatycznie wykorzystywany w momencie tworzenia rezerwacji bądź wypożyczenia
- **VARCHAR** – opisuje większość atrybutów encji, takich jak status rezerwacji, autora, tytuł oraz link do obrazka książki bądź imię, nazwisko, email i hasło klienta.



## **Encje:**

Da się wyróżnić encje w modelu naszej bazy danych przedstawiającej małą bibliotekę:

- Hire
- Reservation
- User
- Book
- BookCopy
- Opinion
- Message

## **Związki między encjami, ich liczebność i uczestnictwo encji:**

W modelu tym mamy styczność z wieloma związkami (relacjami) między encjami, i są to:

### **• Rezerwacje książki**

Opis: Relacja między rezerwacjami a egzemplarzem książki do której są przypisane.

Liczebność: wiele do jednego (N, 1);

Wiele rezerwacji może być przypisanych do jednego egzemplarzu książki ale tylko jeden egzemplarz książki może być przypisana do każdej rezerwacji

Uczestnictwo: Egzemplarz książki - WYMAGANE, rezerwacja - OPCJONALNE

Każda rezerwacja musi mieć przypisaną książkę, ale nie każda książka musi mieć przypisane rezerwacje

### • Rezerwacje klienta

Opis: Relacja pomiędzy rezerwacjami książki a klientem do którego są przypisane

Liczebność: wiele do jednego(N,1)

Wiele rezerwacji może być przypisanych do jednego klienta, ale tylko jeden klient może być przypisany do każdej rezerwacji

Uczestnictwo: Klient - WYMAGANE, rezerwacja - OPCJONALNE

Każda rezerwacja musi mieć przypisanego klienta, ale nie każdy klient musi mieć przypisane rezerwacje.

### • Egzemplarz książki

Opis: Relacja między egzemplarzami książek, a książkami do których są przypisane.

Liczebność: wiele do jednego (N, 1)

Książka ma wiele egzemplarzy, a egzemplarz przynależy do jednej książki.

Uczestnictwo: Egzemplarz - OPCJONALNE, Książka - WYMAGANE

Książka może nie mieć przypisanego egzemplarza, ale każdy egzemplarz przynależy do książki.

### • Opinia książki

Opis: Relacja między opinią, a książką do której jest przypisana

Liczebność: Wiele do jednego (N, 1)

Wiele opinii może być przypisanych do jednej książki ale tylko jedna książka może być przypisana do każdej opinii

Uczestnictwo: Książka - WYMAGANE, opinia - OPCJONALNE

Każda opinia musi mieć przypisaną książkę, ale nie każda książka musi mieć przypisane opinie.

### • Wiadomości klienta

Opis: Relacja między wiadomościami a klientem do którego należą. (wiadomości wysłane do klienta)

Liczebność: wiele do jednego (N, 1)

Wiele wiadomości może być przypisanych do jednego klienta.

Uczestnictwo: wiadomość- OPCJONALNE, klient - WYMAGANE

Każda wiadomość musi mieć przypisanego klienta ale nie każdy klient musi mieć przypisaną wiadomość.

### • Wypożyczenie klienta

Opis: Relacja między wypożyczeniami, a klientem do którego są przypisane.

Liczebność: wiele do jednego (N, 1)

Wiele wypożyczeń może być przypisanych do jednego klienta, ale tylko jeden klient może być przypisany do wypożyczenia

Uczestnictwo: Klient - WYMAGANE, Wypożyczenie - OPCJONALNE

Każde wypożyczenie musi mieć przypisanego klienta, ale nie każdy klient musi mieć przypisane wypożyczenie

### • Wypożyczenie egzemplarzu

Opis: Relacja pomiędzy wypożyczeniem a egzemplarzem do którego jest przypisane

Liczebność: jeden do jednego (N,1)

Każdy egzemplarz może być przypisany do wielu wypożyczeń, ponieważ są one archiwizowane, natomiast każde wypożyczenie może być przypisane tylko do jednego egzemplarza.

Uczestnictwo: Egzemplarz - WYMAGANY, Wypożyczenie - OPCJONALNE

Nie każdy egzemplarz musi posiadać wypożyczenie, ale każde wypożyczenie musi mieć przypisany egzemplarz.

- **Wypożyczenie egzemplarzu**

Opis: Relacja między opinią, a klientem.

Liczebność: wiele do jednego (N,1)

Każdy klient może dodać wiele opinii, a dana opinia może być napisana przez jednego klienta.

Uczestnictwo: Klient - WYMAGANY, Opinia - OPCJONALNE

Nie każdy klient musi dodawać opinie, ale każda opinia musi posiadać autora, czyli klienta.

**Uwarunkowania technologiczne:**

Baza danych będzie posiadała następujące funkcje:

- **Ograniczenia:**

- NOT NULL – oznacza że pole danych nie może być puste
- DEFAULT – wartość domyślna
- CHECK – wymusza sprawdzenie poprawności danych

- **Perspektywy:**

- VIEW – widoki wywiedzione z innych tabeli

- **Procedury przechowywane:**

- PROCEDURE – ułatwiające wielokrotne wykonywanie zaawansowanych zapytań

- **Zaawansowane wyszukiwanie danych:**

- SELECT, ORDER BY, JOIN, LIKE, HAVING, WHERE – umożliwiające wyszukiwanie wg różnych kryteriów;

- **Funkcje agregujące:**

- AVG, COUNT, SUM, MIN, MAX – obliczając średnią arytmetyczną, zliczającą, sumującą, wartości minimalne i maksymalne.

### 3.1.2. Model logiczny i fizyczny

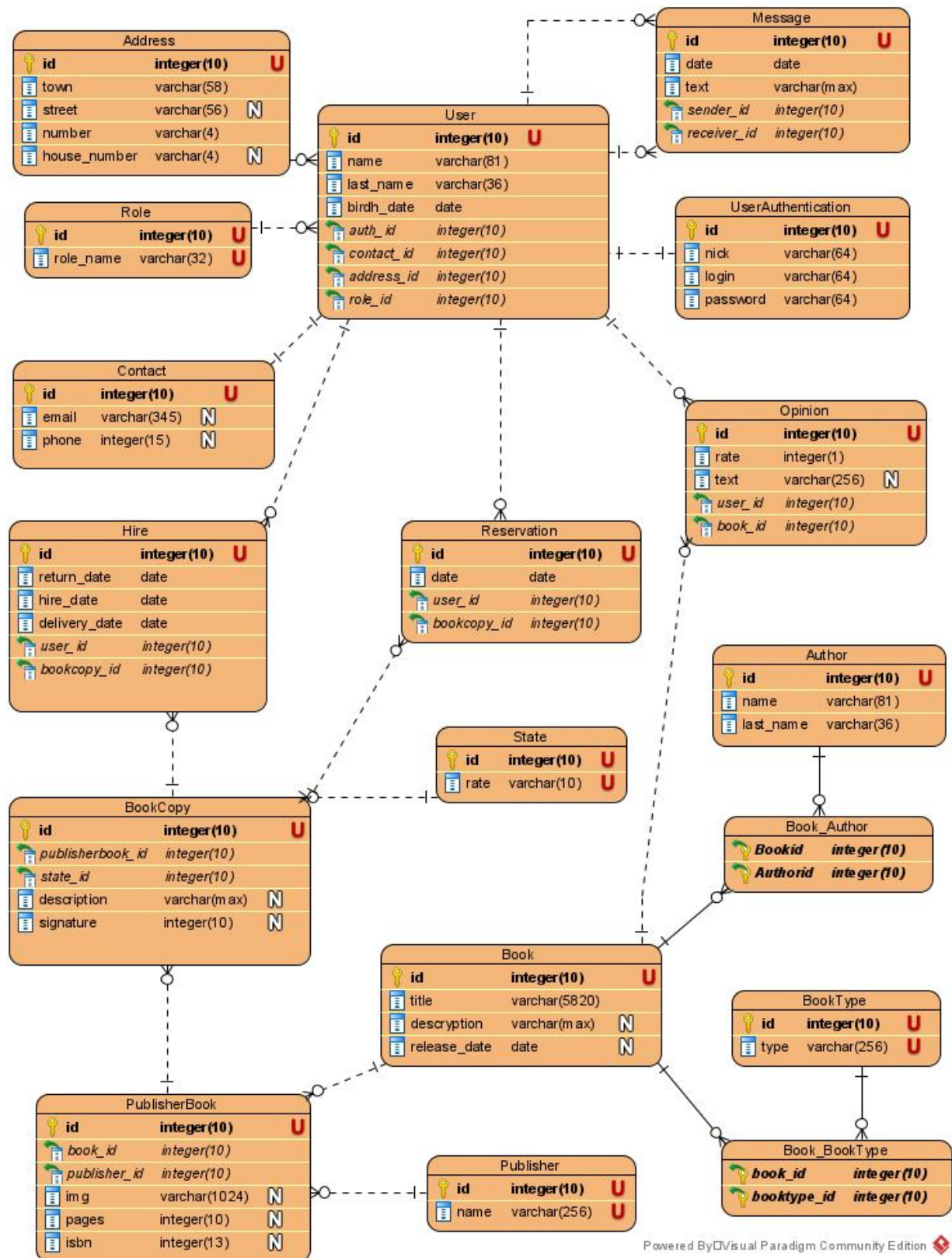


Diagram przedstawia model logiczny i fizyczny jednocześnie.

Opis ograniczeń:

**U** - Unique    **N** - Nullable

### 3.1.3. Mechanizmy bezpieczeństwa

Nasza aplikacja ma zawierać następujące mechanizmy bezpieczeństwa:

- Hashowanie haseł
- Architektura klient-serwer
- Autoryzacja użytkowników
- Rejestrowanie wykonanych operacji (logi)

### 3.1.4. Metoda połączenia z bazą danych

Aplikacja łączy się z bazą danych przy pomocy *Java DataBase Connectivity (JDBC)*.

### 3.1.5. Triggery

Jeśli chodzi o ten element postanowiliśmy z niego zrezygnować gdyż stwierdziliśmy że nasz system nie wymaga tego typu elementów. Jeśli zajdzie potrzeba realizacji tego typu zdarzeń ( na przykład logi ) to zrealizujemy je bezpośrednio w oprogramowaniu.

### 3.1.6. Indeksy

Indeksowanie najlepiej zastosować dla tabel, których dane są często poszukiwane. Indeksy działają mało optymalnie gdy dane tabeli są często dodawane/usuwane lub modyfikowane.

Na podstawie tych kryteriów wytypowano następujące indeksy:

- Tabela User:
  - imię i nazwisko klienta
- Tabela Book:
  - tytuły książek
- Tabela Wypożyczenia:
  - id klienta
  - id egzemplarza
- Tabela Rezerwacje:
  - id klienta
  - id książki

Imię i Nazwisko klienta prawie nigdy nie następuje zmianie. Dodatkowo nie często dodawany jest do bazy danych nowy klient. Podobnie tyczy się to książek i ich tytułów.

### 3.1.7. Widoki

- Książki dostępne do wypożyczenia:

Widok przedstawia książki, które mają przypisany co najmniej jeden egzemplarz bez przypisanego wypożyczenia.

Widok zwraca tytuł książki (title, tabela Book), oraz liczba egzemplarzy bez przypisanych wypożyczeń.

- Klienci którzy posiadają książki z przekroczonym terminem

Widok przedstawia klientów, do których przypisane są wypożyczenia, w których data return\_date poprzedza aktualną datę. Dodatkowo książki te muszą nie być jeszcze zwrócone - nie mają przypisanego delivery\_date.

Widok zwraca Imiona i Nazwiska klientów (name, last\_name tabela User), oraz ilość dni opóźnienia oddania każdej z książek, oraz tytuły tych książek (title tabela Book).

- Wypożyczenia klienta

Widok przedstawia klientów oraz tytuły książek powiązane z ich wypożyczeniami.

Widok zwraca imię oraz nazwisko klienta (name, last\_name tabela User) oraz tytuły książek (title tabela Book), powiązane z egzemplarzami wypożyczonych książek.

- Rezerwacje klienta

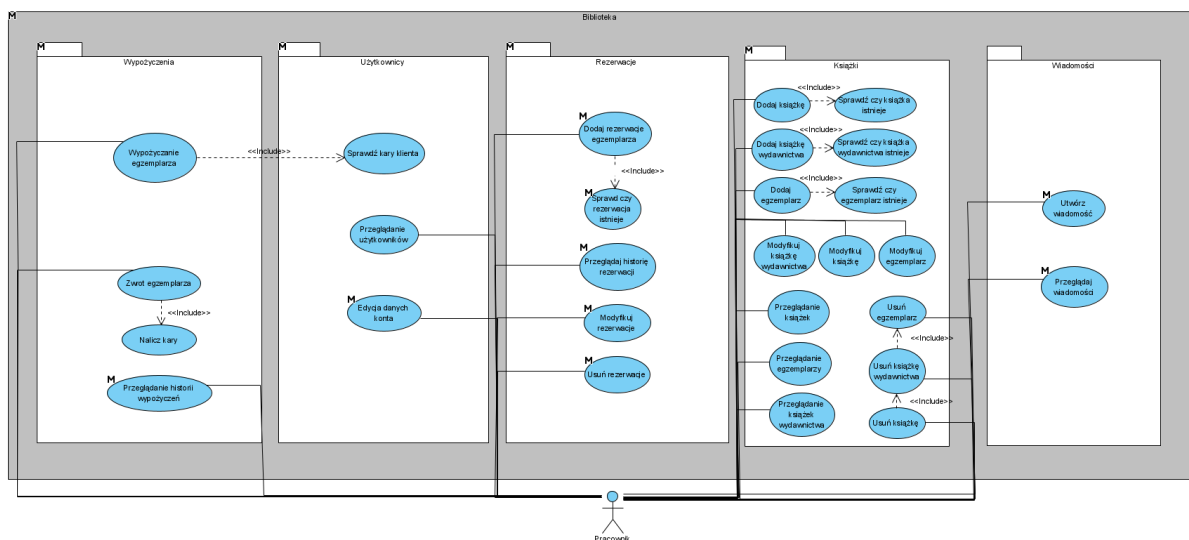
Widok przedstawia klientów oraz tytuły książek powiązane z ich rezerwacjami.

Widok zwraca imię oraz nazwisko klienta (name, last\_name tabela User) oraz tytuły książek (title tabela Book), powiązane z rezerwacjami książek.

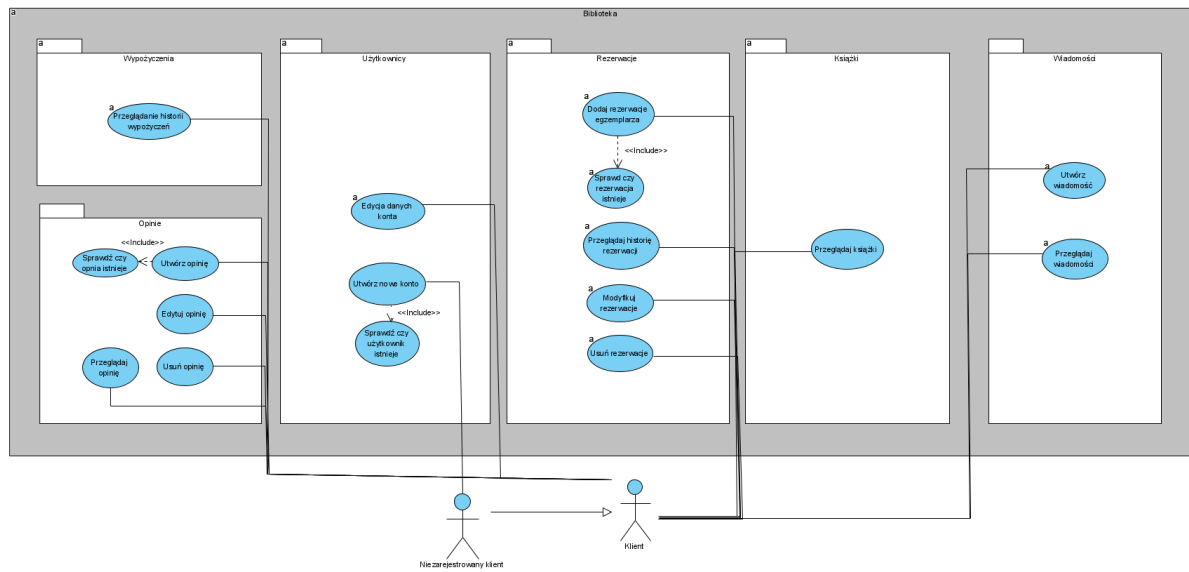
## 3.2. Projekt aplikacji użytkownika

### 3.2.1. Diagram przypadków użycia

#### 3.2.1.1 Diagram przypadków użycia dla pracownika



### 3.2.1.2 Diagram przypadków użycia dla klienta






## 3.2.2 Interfejs graficzny i struktura menu

### 3.2.2.1 Przykładowe menu główne

Wyloguj się



# Księgarnia publiczna im. Bolesława Prusa

np. tytuł, imię autora, gatunek itd...

Szukaj


Książki

Użytkownicy

Wypożyczenia i Rezerwacje

Wiadomości


## Najpopularniejsze tytuły



Bolesław Prus

**Lalka**


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut iaculis nisi et urna commodo congue. Phasellus efficitur libero sed nulla.



Bolesław Prus

**Lalka**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut iaculis nisi et urna commodo congue. Phasellus efficitur libero sed nulla.




Bolesław Prus

**Lalka**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut iaculis nisi et urna commodo congue. Phasellus efficitur libero sed nulla.

[Pokaż więcej](#)


## Najlepiej oceniane



Bolesław Prus

**Lalka**


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut iaculis nisi et urna commodo congue. Phasellus efficitur libero sed nulla.



Bolesław Prus

**Lalka**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut iaculis nisi et urna commodo congue. Phasellus efficitur libero sed nulla.



Bolesław Prus

**Lalka**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut iaculis nisi et urna commodo congue. Phasellus efficitur libero sed nulla.

[Pokaż więcej](#)

### 3.2.2.2 Przykładowe pole do wysyłania wiadomości

NAPISZ WIADOMOŚĆ DO UŻYTKOWNIKA

Nick użytkownika:

Treść wiadomości

Wyślij wiadomość

Anuluj

### 3.2.2.3 Menu wyboru wiadomości

Odebrane

Odbierz wiadomości

Wysłane

wiadomość 1

wiadomość 2

wiadomość 3

wiadomość 4

wiadomość 5

wiadomość 6

wiadomość 7

wiadomość 8

wiadomość 9

Wyślij wiadomość

### 3.2.2.4 Tworzenie i edycja nowej książki

tytuł			
opis			
autor			
Data wydania:			
<input type="text" value="dd"/>	<input type="text" value="mm"/>	<input type="text" value="rrrr"/>	
	Egzemplarz 1 <a href="#">edytuj</a>		Wydanie 1 <a href="#">edytuj</a>
	Egzemplarz 2 <a href="#">edytuj</a>		Wydanie 2 <a href="#">edytuj</a>
	Dodaj nowy egzemplarz		Dodaj nowe wydanie
Zapisz			

-

## 4. Implementacja systemu bazy danych

Baza danych została zaimplementowana w systemie MySQL. W celach testowych baza danych została utworzona przy pomocy strony [ServerProject.pl](http://ServerProject.pl). Podstawowa konfiguracja tam była bardzo prosta, strona utworzyła dla nas bazę danych z odpowiednimi danymi dostępowymi. Za pomocą aplikacji PhpMyAdmin wykonaliśmy zapytania do bazy danych.

### 4.1. Tworzenie tabel i definiowanie ograniczeń

#### 4.1.1. Komentarz do zastosowanych rozwiązań

Tabele zostały utworzone na bazie modelu logiczno-fizycznego. Elementy składowe tabel w pełni pokrywają się z elementami modelu.

W trakcie pisania kodu dokonaliśmy jedynie drobnych zmian wynikających czysto z możliwości MySQL. Jedną z nich była rezygnacja z Varchar(MAX). Zamiast tego korzystaliśmy z dużej liczby, która powinna być wystarczająca do przechowywania danych nawet w ekstremalnych przypadkach, rozmiar ten wynosi 10 000 znaków.

Dodatkowo number oraz user były już nazwami zajętymi w przestrzeni MySQL. Aby nie naruszać spójności semantycznej modeli z kodem, dodany został znak podłogi na początku owych nazw.

#### 4.1.2 Kod Tabel

```
CREATE TABLE Address(  
id integer(10) NOT NULL UNIQUE AUTO_INCREMENT,  
town varchar(58) NOT NULL,  
street varchar(58),  
_number varchar(58) NOT NULL,  
house_number varchar(4),  
PRIMARY KEY (id)  
);
```

```
CREATE TABLE Role(  
id integer(10) NOT NULL UNIQUE AUTO_INCREMENT,  
role_name varchar(32) NOT NULL UNIQUE  
);
```

```
CREATE TABLE Contact(  
id integer(10) NOT NULL UNIQUE AUTO_INCREMENT,  
email varchar(345),  
phone varchar(15),  
PRIMARY KEY (id)  
);
```

```

CREATE TABLE UserAuthentication(
    id int(10) NOT NULL UNIQUE AUTO_INCREMENT,
    nick varchar(64) NOT NULL,
    login varchar(64) NOT NULL,
    password varchar(64) NOT NULL,
    PRIMARY KEY(id)
);

CREATE TABLE Publisher (
    id int(10) NOT NULL UNIQUE AUTO_INCREMENT,
    name varchar(256) NOT NULL UNIQUE
);

CREATE TABLE BookType(
    id int(10) NOT NULL UNIQUE AUTO_INCREMENT,
    type varchar(256) NOT NULL UNIQUE,
    PRIMARY KEY(id)
);

CREATE TABLE Author(
    id int(10) NOT NULL UNIQUE AUTO_INCREMENT,
    name varchar(81) NOT NULL,
    last_name varchar(36) NOT NULL,
    PRIMARY KEY(id)
);

CREATE TABLE State (
    id int(10) NOT NULL UNIQUE AUTO_INCREMENT,
    rate varchar(10) NOT NULL UNIQUE
);

CREATE TABLE Book (
    id int(10) NOT NULL UNIQUE AUTO_INCREMENT,
    title varchar(5820) NOT NULL,
    description varchar(10000),
    release_date date,
    PRIMARY KEY (id)
);

CREATE TABLE Book_Author(
    book_id int(10) NOT NULL,
    author_id int(10) NOT NULL,
    FOREIGN KEY(book_id) REFERENCES Book(id),
    FOREIGN KEY(author_id) REFERENCES Author(id)
);

CREATE TABLE Book_BookType(

```

```

    book_id int(10) NOT NULL,
    booktype_id int(10) NOT NULL,
    FOREIGN KEY(book_id) REFERENCES Book(id),
    FOREIGN KEY(booktype_id) REFERENCES BookType(id)
);

```

```

CREATE TABLE PublisherBook(
id integer(10) NOT NULL UNIQUE AUTO_INCREMENT ,
book_id integer(10) NOT NULL,
publisher_id integer(10) NOT NULL,
img varchar(10000),
pages integer(10),
isbn integer(13),
PRIMARY KEY (id),
FOREIGN KEY(book_id) REFERENCES Book(id),
FOREIGN KEY(publisher_id) REFERENCES Publisher(id)
);

```

```

CREATE TABLE BookCopy(
id integer(10) NOT NULL UNIQUE AUTO_INCREMENT ,
publisherbook_id integer(10) NOT NULL,
state_id integer(10) NOT NULL,
description varchar(10000),
signature integer(10),
PRIMARY KEY (id),
FOREIGN KEY(publisherbook_id) REFERENCES PublisherBook(id),
FOREIGN KEY(state_id) REFERENCES State(id)
);

```

```

CREATE TABLE _User(
    id int(10) NOT NULL UNIQUE AUTO_INCREMENT ,
    name varchar(81) NOT NULL,
    last_name varchar(36) NOT NULL,
    birth_date date NOT NULL,
    auth_id int(10) NOT NULL,
    contact_id int(10) NOT NULL,
    address_id int(10) NOT NULL,
    role_id int(10) NOT NULL,
    PRIMARY KEY (id),
    FOREIGN KEY(auth_id) REFERENCES UserAuthentication(id),
    FOREIGN KEY(contact_id) REFERENCES Contact(id),
    FOREIGN KEY(address_id) REFERENCES Address(id),
    FOREIGN KEY(role_id) REFERENCES Role(id)
);

```

```

CREATE TABLE Message(
    id int(10) NOT NULL UNIQUE AUTO_INCREMENT ,
    date date NOT NULL,

```

```

text varchar(10000) NOT NULL,
sender_id int(10) NOT NULL,
receiver_id int(10) NOT NULL,
PRIMARY KEY(id),
FOREIGN KEY(sender_id) REFERENCES _User(id),
FOREIGN KEY(receiver_id) REFERENCES _User(id)
);

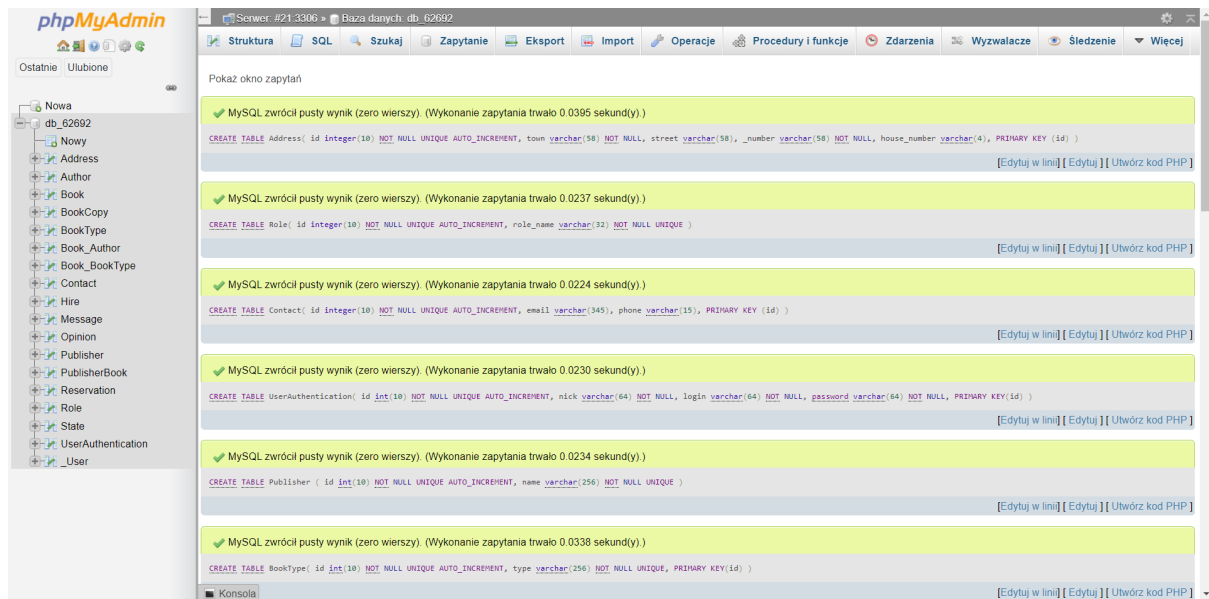
CREATE TABLE Reservation (
    id int(10) NOT NULL UNIQUE AUTO_INCREMENT,
    _date date NOT NULL,
    user_id int(10) NOT NULL,
    bookcopy_id int(10) NOT NULL,
    PRIMARY KEY (id),
    FOREIGN KEY(user_id) REFERENCES _User(id),
    FOREIGN KEY(bookcopy_id) REFERENCES BookCopy(id)
);

CREATE TABLE Hire(
    id integer(10) NOT NULL UNIQUE AUTO_INCREMENT,
    return_date date NOT NULL,
    hire_date date NOT NULL,
    delivery_date date,
    user_id integer(10) NOT NULL,
    bookcopy_id integer(10) NOT NULL,
    PRIMARY KEY (id),
    FOREIGN KEY(user_id) REFERENCES _User(id),
    FOREIGN KEY(bookcopy_id) REFERENCES BookCopy(id)
);

CREATE TABLE Opinion(
    id int(10) NOT NULL UNIQUE AUTO_INCREMENT,
    rate int(1) NOT NULL,
    text varchar(256),
    user_id int(10) NOT NULL,
    book_id int(10) NOT NULL,
    PRIMARY KEY(id),
    FOREIGN KEY(user_id) REFERENCES _User(id),
    FOREIGN KEY(book_id) REFERENCES Book(id)
);

```





Rysunek 1. Wynik dodatnia wszystkich tabel

### 4.1.3. Problemy napotkane przy definiowaniu tabel

#### Błąd

##### Analiza statyczna:

2 błędów zostało znalezionych podczas analizy.

1. Ta opcja zawiera konflikt z "TABLE". (near "User" at position 13)
2. Oczekiwano nazwy obiektu. (near "(" at position 17)

##### Zapytanie SQL: [Kopiuuj](#)

```
CREATE TABLE User( id int(10) NOT NULL UNIQUE AUTO_INCREMENT, name varchar(81) NOT NULL, last_name varchar(36) NOT NULL, birth_date date NOT NULL, auth_id int(10) NOT NULL, contact_id int(10) NOT NULL, address_id int(10) NOT NULL, role_id int(10) NOT NULL, PRIMARY KEY (id), FOREIGN KEY(auth_id) REFERENCES UserAuthentication(id), FOREIGN KEY(contact_id) REFERENCES Contact(id), FOREIGN KEY(address_id) REFERENCES Address(id), FOREIGN KEY(role_id) REFERENCES Role(id) )
```

##### MySQL zwrócił komunikat: [?](#)

#1005 - Nie można stworzyć tabeli `db\_62692`.`User` (Kod błędu: 150 "Foreign key constraint is incorrectly formed") ([Szczegóły...](#))

Rysunek 2. Błąd napotkany przy próbie zdefiniowania tabeli o nazwie User

Podobny problem jak na *Rysunek2* wystąpił również przy próbie nazwania kolumny nazwą "date". Dla bazy danych MySQL *date* to typ danych więc również nam na to nie pozwolił. Aby rozwiązać ten problem na początku słów które powodują sprzeczności dodaliśmy podłogę np. `_User`.

#### Błąd

##### Zapytanie SQL: [Kopiuuj](#)

```
CREATE TABLE Opinion(
  id int(10) NOT NULL UNIQUE AUTO_INCREMENT,
  rate int(1) NOT NULL,
  text varchar(256),
  user_id int(10) NOT NULL,
  book_id int(10) NOT NULL,
  ...
```

##### MySQL zwrócił komunikat: [?](#)

#1005 - Nie można stworzyć tabeli `db\_62692`.`Opinion` (Kod błędu: 150 "Foreign key constraint is incorrectly formed") ([Szczegóły...](#))

Rysunek 3. Błąd napotkany przy przy złej kolejności definiowania tabel

## 4.2. Implementacja mechanizmów przetwarzania danych

### 4.2.1. Indeksy

```
CREATE INDEX _User_index1  
on _User (name, last_name);
```

```
CREATE INDEX Book_index1  
on Book (title);
```

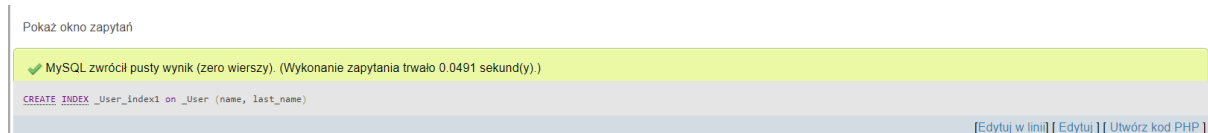
```
CREATE INDEX Hire_index1  
on Hire (user_id);
```

```
CREATE INDEX Hire_index2  
on Hire (bookcopy_id);
```

```
CREATE INDEX Hire_index3  
on Hire (delivery_date dsc);
```

```
CREATE INDEX Reservation_index1  
on Reservation (user_id);
```

```
CREATE INDEX Reservation_index2  
on Reservation (bookcopy_id);
```



Rysunek4. Dodanie indeksu \_User\_index1

### 4.2.2. Widoki

Widok "ksiazki\_na\_stanie" przedstawia książki, które użytkownik może wypożyczyć. Zwraca on tytuł książki oraz liczbę egzemplarzy.

```
CREATE VIEW ksiazki_na_stanie AS  
SELECT Book.title,COUNT(*) FROM BookCopy  
INNER JOIN PublisherBook ON BookCopy.publisherbook_id=PublisherBook.id  
INNER JOIN Book ON PublisherBook.book_id=Book.id  
GROUP BY Book.title;
```

Widok "wypozyczenia\_klienta" prezentuje klientów i tytuły książek, powiązanych z ich wypożyczeniami. Zwraca imię i nazwisko klienta oraz tytuł książki.

```
CREATE VIEW wypozyczenia_klienta AS  
SELECT _User.name, _User.last_name, Book.title FROM Hire  
INNER JOIN _User ON Hire.user_id=_User.id
```

```
INNER JOIN BookCopy ON Hire.bookcopy_id=BookCopy.id
INNER JOIN PublisherBook ON BookCopy.publisherbook_id=PublisherBook.id
INNER JOIN Book ON PublisherBook.book_id=Book.id
WHERE Hire.delivery_date IS NULL;
```

Widok “nieoddane\_książki” przedstawia klientów, którzy zalegają z oddaniem książki lub książek. Zwraca imiona i nazwiska klientów oraz ilość dni opóźnienia, w oddaniu książki.

```
CREATE VIEW nieoddane_książki AS SELECT name, last_name, return_date, title
FROM Hire
INNER JOIN _User ON Hire.user_id = _User.id
INNER JOIN BookCopy ON Hire.bookcopy_id = BookCopy.id
INNER JOIN PublisherBook ON BookCopy.publisherbook_id = PublisherBook.id
INNER JOIN Book ON PublisherBook.book_id = Book.id
WHERE return_date < (SELECT CURDATE()) AND
(delivery_date IS NULL);
```

Widok “rezerwacje” przedstawia klientów oraz tytuły książek, które są przez nich zarezerwowane. Imię, nazwisko i tytuły książek zostają zwrócone.

```
CREATE VIEW rezerwacje AS SELECT name, last_name, title
FROM Reservation
INNER JOIN _User ON Reservation.user_id = _User.id
INNER JOIN BookCopy ON Reservation.bookcopy_id = BookCopy.id
INNER JOIN PublisherBook ON BookCopy.publisherbook_id = PublisherBook.id
INNER JOIN Book ON PublisherBook.book_id = Book.id
ORDER BY last_name;
```

Pokaż okno zapytań

✓ MySQL zwrócił pusty wynik (zero wierszy). (Wykonanie zapytania trwało 0.0197 sekund(y).)

```
CREATE VIEW dostepne_książki AS SELECT Book.title,COUNT(*) FROM BookCopy INNER JOIN PublisherBook ON BookCopy.publisherbook_id=PublisherBook.id INNER JOIN Book ON PublisherBook.book_id=Book.id GROUP BY Book.title
```

[Edytuj w linii] [Edytuj] [Utwórz kod PHP]

✓ MySQL zwrócił pusty wynik (zero wierszy). (Wykonanie zapytania trwało 0.0083 sekund(y).)

```
CREATE VIEW wypożyczenia_klienta AS SELECT _User.name, _User.last_name, Book.title FROM Hire INNER JOIN _User ON Hire.user_id=_User.id INNER JOIN BookCopy ON Hire.bookcopy_id=BookCopy.id INNER JOIN PublisherBook ON BookCopy.publisherbook_id=PublisherBook.id INNER JOIN Book ON PublisherBook.book_id=Book.id WHERE Hire.delivery_date IS NULL
```

[Edytuj w linii] [Edytuj] [Utwórz kod PHP]

✓ MySQL zwrócił pusty wynik (zero wierszy). (Wykonanie zapytania trwało 0.0225 sekund(y).)

```
CREATE VIEW nieoddane_książki AS SELECT name, last_name, return_date, title FROM Hire INNER JOIN _User ON Hire.user_id = _User.id INNER JOIN BookCopy ON Hire.bookcopy_id = BookCopy.id INNER JOIN PublisherBook ON BookCopy.publisherbook_id = PublisherBook.id INNER JOIN Book ON PublisherBook.book_id = Book.id WHERE return_date < (SELECT CURDATE()) AND (delivery_date IS NULL)
```

[Edytuj w linii] [Edytuj] [Utwórz kod PHP]

✓ MySQL zwrócił pusty wynik (zero wierszy). (Wykonanie zapytania trwało 0.0093 sekund(y).)

```
CREATE VIEW rezerwacje AS SELECT name, last_name, title FROM Reservation INNER JOIN _User ON Reservation.user_id = _User.id INNER JOIN BookCopy ON Reservation.bookcopy_id = BookCopy.id INNER JOIN PublisherBook ON BookCopy.publisherbook_id = PublisherBook.id INNER JOIN Book ON PublisherBook.book_id = Book.id ORDER BY last_name
```

[Edytuj w linii] [Edytuj] [Utwórz kod PHP]

Rysunek 5. Wynik dodania widoku

### 4.3. Testowanie bazy danych na przykładowych danych

Poniższe kody dodają do bazy danych przykładowe dane testowe.

```
INSERT INTO Address( id, town, street, _number, house_number)
VALUES (1, 'Warszawa', 'Lipowa', '24', '16');
INSERT INTO Address( id, town, street, _number, house_number)
VALUES (2, 'Gdynia', 'Azaliowa', '10', NULL);
INSERT INTO Address( id, town, street, _number, house_number)
VALUES (3, 'Warszawa', 'Lipowa', '5', NULL);
INSERT INTO Address( id, town, street, _number, house_number)
VALUES (4, 'Warszawa', 'Smutna', '18', '22');
INSERT INTO Address( id, town, street, _number, house_number)
VALUES (5, 'Gdynia', 'Radosna', '99', NULL);
INSERT INTO Address( id, town, street, _number, house_number)
VALUES (6, 'Bydgoszcz', 'Ziemniaczana', '1', NULL);
INSERT INTO Address( id, town, street, _number, house_number)
VALUES (7, 'Legnica', 'Targowicka', '4', '3');
INSERT INTO Address( id, town, street, _number, house_number)
VALUES (8, 'Opole', 'Orla', '3', NULL);
INSERT INTO Address( id, town, street, _number, house_number)
VALUES (9, 'Tychy', 'Lisia', '12', '2');
INSERT INTO Address( id, town, street, _number, house_number)
VALUES (10, 'Sosnowiec', 'Ptasia', '5', NULL);
```

```
INSERT INTO Role( id, role_name)
VALUES (1, 'pracownik');
INSERT INTO Role( id, role_name)
VALUES (2, 'klient');
```

```
INSERT INTO Contact(id, email, phone)
VALUES ( 1, 'malyjasio@wp.pl' ,48605554856 );
INSERT INTO Contact(id, email, phone)
VALUES ( 2,NULL ,NULL );
INSERT INTO Contact(id, email, phone)
VALUES ( 3, 'turboczytelnik@gmail.com' ,NULL );
INSERT INTO Contact(id, email, phone)
VALUES ( 4,NULL ,48505554900 );
INSERT INTO Contact(id, email, phone)
VALUES ( 5, 'barbara001@gmail.com' ,48455552789 );
```

```
INSERT INTO UserAuthentication(id, nick, login, password)
VALUES (1, 'Zbychu', 'Zbychu', '1234');
INSERT INTO UserAuthentication(id, nick, login, password)
VALUES (2, 'kwiatuszek01', 'amelia01', 'Hc6ZD8663lOp');
INSERT INTO UserAuthentication(id, nick, login, password)
VALUES (3, 'Zielone wody', 'bartek007', 'Kotekliżemleczo');
INSERT INTO UserAuthentication(id, nick, login, password)
```

```

VALUES (4, 'aeopjasdwah', 'vilette444', 'Mojehaslo321');
INSERT INTO UserAuthentication(id, nick, login, password)
VALUES (5, 'brokencandle', 'cancersandcrabs', 'LOLOLO');
INSERT INTO Publisher( id, name)
VALUES (1, 'Greg' );
INSERT INTO Publisher( id, name)
VALUES (2, 'Albatros' );
INSERT INTO Publisher( id, name)
VALUES (3, 'Drzewo Babel' );
INSERT INTO Publisher( id, name)
VALUES (4, 'ELAY' );
INSERT INTO Publisher( id, name)
VALUES (5, 'Fabryka Słów' );

INSERT INTO BookType( id, type)
VALUES (1, 'literatura obyczajowa');
INSERT INTO BookType( id, type)
VALUES (2, 'romans');
INSERT INTO BookType( id, type)
VALUES (3, 'kryminał');
INSERT INTO BookType( id, type)
VALUES (4, 'sensacja');
INSERT INTO BookType( id, type)
VALUES (5, 'thriller');
INSERT INTO BookType( id, type)
VALUES (6, 'fantastyka');
INSERT INTO BookType( id, type)
VALUES (7, 'science fiction');
INSERT INTO BookType( id, type)
VALUES (8, 'literatura faktu');
INSERT INTO BookType( id, type)
VALUES (9, 'horror');
INSERT INTO BookType( id, type)
VALUES (10, 'literatura młodzieżowa');

INSERT INTO Author VALUES (1, 'Adam', 'Żuk');
INSERT INTO Author VALUES (2, 'Nicola', 'Tesla');
INSERT INTO Author VALUES (3, 'Albert', 'Einstein');
INSERT INTO Author VALUES (4, 'Elon', 'Musk');
INSERT INTO Author VALUES (5, 'Rhoshandiatellyneshiaunneveshenk', 'Williams');

INSERT INTO State VALUES (1, 'zły');
INSERT INTO State VALUES (2, 'średni');
INSERT INTO State VALUES (3, 'dobry');
INSERT INTO State VALUES (4, 'bardzo dobry');
INSERT INTO State VALUES (5, 'nowy');

INSERT INTO Book VALUES (1, 'Pan Tadeusz', 'fajna książka', '1000-01-01');

```

```

INSERT INTO Book VALUES (2,'Pan Pomidor',NULL,NULL);
INSERT INTO Book VALUES (3,'Ferdydurke','Gombrowicz to geniusz','1001-01-01');
INSERT INTO Book VALUES (4,'Krzyżacy','fajna książka','1001-01-10');
INSERT INTO Book VALUES (5,'Lalka','fajna książka','1001-01-11');

```

```

INSERT INTO Book_Author VALUES (1,1);
INSERT INTO Book_Author VALUES (2,2);
INSERT INTO Book_Author VALUES (3,3);
INSERT INTO Book_Author VALUES (4,4);
INSERT INTO Book_Author VALUES (5,5);

```

```

INSERT INTO Book_BookType VALUES (1,1);
INSERT INTO Book_BookType VALUES (2,2);
INSERT INTO Book_BookType VALUES (3,3);
INSERT INTO Book_BookType VALUES (4,4);
INSERT INTO Book_BookType VALUES (5,5);

```

```

INSERT INTO PublisherBook VALUES
(1,1,1,'https://static4.redcart.pl/templates/images/thumb/7435/1500/1500/pl/0/templates/images/products/7435/526066f185eaac903d5d36fd4fbf83a7.jpg',10,10);
INSERT INTO PublisherBook VALUES
(2,2,2,'https://static4.redcart.pl/templates/images/thumb/7435/1500/1500/pl/0/templates/images/products/7435/526066f185eaac903d5d36fd4fbf83a7.jpg',NULL,10);
INSERT INTO PublisherBook VALUES
(3,3,3,'https://static4.redcart.pl/templates/images/thumb/7435/1500/1500/pl/0/templates/images/products/7435/526066f185eaac903d5d36fd4fbf83a7.jpg',10,NULL);
INSERT INTO PublisherBook VALUES
(4,4,4,'https://static4.redcart.pl/templates/images/thumb/7435/1500/1500/pl/0/templates/images/products/7435/526066f185eaac903d5d36fd4fbf83a7.jpg',10,100);
INSERT INTO PublisherBook VALUES
(5,5,5,'https://static4.redcart.pl/templates/images/thumb/7435/1500/1500/pl/0/templates/images/products/7435/526066f185eaac903d5d36fd4fbf83a7.jpg',1000,10);

```

```

INSERT INTO BookCopy(id, publisherbook_id, state_id, description, signature) VALUES(1,
1, 1, 'opis1', 'Kowalski');
INSERT INTO BookCopy(id, publisherbook_id, state_id, description, signature) VALUES(2,
2, 2, 'opis2', 'Nowak');
INSERT INTO BookCopy(id, publisherbook_id, state_id, description, signature) VALUES(3,
3, 3, 'opis3', 'Brzeczyszczykiewicz');
INSERT INTO BookCopy(id, publisherbook_id, state_id, description, signature) VALUES(4,
4, 4, 'opis4', 'Łuczak');
INSERT INTO BookCopy(id, publisherbook_id, state_id, description, signature) VALUES(5,
5, 5, 'opis5', 'Piłsudski');

```

```

INSERT INTO _User VALUES(1, 'Michał', 'Miller', '1900-07-19', 1, 1, 1, 1);
INSERT INTO _User VALUES(2, 'Maciej', 'Fuks', '1200-02-19', 2, 2, 2, 2);
INSERT INTO _User VALUES(3, 'Anna', 'Lepioł', '1300-03-19', 3, 3, 3, 2);
INSERT INTO _User VALUES(4, 'Marcin', 'Korzeniowski', '2002-04-20', 4, 4, 4, 2);

```

```
INSERT INTO _User VALUES(5, 'Agnieszka', 'Dygant', '2009-10-20', 5, 5, 5, 2);
```

```
INSERT INTO Message(id, date, text, sender_id, receiver_id) VALUES(1, '1999-01-19',  
'text1', 1, 1);
```

```
INSERT INTO Message(id, date, text, sender_id, receiver_id) VALUES(2, '1999-02-19',  
'text2', 2, 2);
```

```
INSERT INTO Message(id, date, text, sender_id, receiver_id) VALUES(3, '1999-03-19',  
'text3', 3, 3);
```

```
INSERT INTO Message(id, date, text, sender_id, receiver_id) VALUES(4, '1999-04-19',  
'text4', 4, 4);
```

```
INSERT INTO Message(id, date, text, sender_id, receiver_id) VALUES(5, '2000-05-19',  
'text5', 5, 5);
```

```
INSERT INTO Reservation(id, _date, user_id, bookcopy_id) VALUES(1, '1987-01-19', 1, 1);
```

```
INSERT INTO Reservation(id, _date, user_id, bookcopy_id) VALUES(2, '1234-02-19', 2, 2);
```

```
INSERT INTO Reservation(id, _date, user_id, bookcopy_id) VALUES(3, '1342-03-19', 3, 3);
```

```
INSERT INTO Reservation(id, _date, user_id, bookcopy_id) VALUES(4, '1643-04-19', 4, 4);
```

```
INSERT INTO Reservation(id, _date, user_id, bookcopy_id) VALUES(5, '1743-05-19', 5, 5);
```

```
INSERT INTO Hire(id, return_date, hire_date, delivery_date, user_id, bookcopy_id)  
VALUES(1, '2021-05-19', '2020-01-19', '1212-11-20', 1, 1);
```

```
INSERT INTO Hire(id, return_date, hire_date, delivery_date, user_id, bookcopy_id)  
VALUES(2, '2029-06-19', '2020-12-19', NULL, 2, 2);
```

```
INSERT INTO Hire(id, return_date, hire_date, delivery_date, user_id, bookcopy_id)  
VALUES(3, '2021-07-20', '2020-11-19', '1222-09-19', 3, 3);
```

```
INSERT INTO Hire(id, return_date, hire_date, delivery_date, user_id, bookcopy_id)  
VALUES(4, '2021-08-20', '2020-12-19', '1333-05-20', 4, 4);
```

```
INSERT INTO Hire(id, return_date, hire_date, delivery_date, user_id, bookcopy_id)  
VALUES(5, '2021-09-20', '2020-01-19', NULL, 5, 5);
```

```
INSERT INTO Opinion(id, rate, text, user_id, book_id) VALUES(1, 1, 'text1', 1, 1);
```

```
INSERT INTO Opinion(id, rate, text, user_id, book_id) VALUES(2, 2, 'text2', 2, 2);
```

```
INSERT INTO Opinion(id, rate, text, user_id, book_id) VALUES(3, 3, 'text3', 3, 3);
```

```
INSERT INTO Opinion(id, rate, text, user_id, book_id) VALUES(4, 4, 'text4', 4, 4);
```

```
INSERT INTO Opinion(id, rate, text, user_id, book_id) VALUES(5, 5, 'text5', 5, 5);
```

#### 4.3.1. Test widoków na przykładowych danych

+ Opcje

 					title	COUNT(*)		
<input type="checkbox"/>		Edytuj		Kopiuj		Usuń	Ferdydurke	1
<input type="checkbox"/>		Edytuj		Kopiuj		Usuń	Krzyżacy	1
<input type="checkbox"/>		Edytuj		Kopiuj		Usuń	Lalka	1
<input type="checkbox"/>		Edytuj		Kopiuj		Usuń	Pan Pomidor	1
<input type="checkbox"/>		Edytuj		Kopiuj		Usuń	Pan Tadeusz	1







Rysunek 6. Wynik wykonania widoku "dostepne\_ksiazki"

<div><div><div></div><div></div><div></div></div></div>				name	last_name	title	
<input type="checkbox"/>		Edytuj	 	Kopiuj	Usuń	Agnieszka Dygant	Lalka
<input type="checkbox"/>		Edytuj	 	Kopiuj	Usuń	Maciej Fuks	Pan Pomidor
<input type="checkbox"/>		Edytuj	 	Kopiuj	Usuń	Marcin Korzeniowski	Krzyżacy
<input type="checkbox"/>		Edytuj	 	Kopiuj	Usuń	Anna Lepioł	Ferdydurke
<input type="checkbox"/>		Edytuj	 	Kopiuj	Usuń	Michał Miller	Pan Tadeusz

Rysunek 7. Wynik wykonania widoku "wypozyczenia\_klienta"

<div><div><div>←</div><div>T</div><div>→</div></div><div>▼</div></div>				name	last_name	return_date	title
<div><div><div><div></div></div></div><div><div><div></div></div></div></div>	<div><div><div></div></div></div> Edytuj	<div><div><div></div></div></div> Kopiuj	<div><div><div></div></div></div> Usuń	Michał Miller	2019-05-19	Pan Tadeusz	

Rysunek 8. Wynik wykonania widoku "nieoddane\_ksiazki"

<div><div><div><div></div><div></div><div></div></div><div></div></div></div>							name	last_name	title
<input type="checkbox"/>		Edytuj		Kopiuj		Usuń	Maciej	Fuks	Pan Pomidor
<input type="checkbox"/>		Edytuj		Kopiuj		Usuń	Agnieszka	Dygant	Lalka

Rysunek 9. Wynik wykonania widoku "rezerwacje"

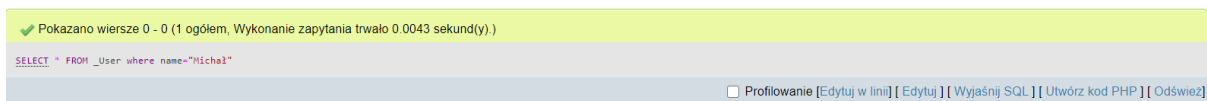
#### 4.3.2. Testy wydajnościowe indeksu

Test wykonaliśmy na indeksie `_User_index1` na zapytaniu:

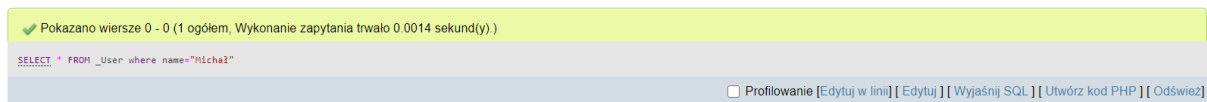
```
SELECT * FROM _User WHERE name = "Michał"
```

Testy zostały wykonane dla tabel o 5 tysięcy wierszach.





Rysunek 10. Czas zwrotu bez indeksu



Rysunek 11. Czas zwrotu z indeksem

#### 4.3.4. Napotkane problemy przy testowaniu

Jak zauważyliśmy MySQL nie obsługuje domyślnie transakcji. Wywnioskowaliśmy to przy wykonywaniu insertów. To błędnie napisane zostały odrzucone, ale inserty napisane poprawnie zostały zrealizowane, co nie powinno się zdarzyć, ponieważ system powinien odrzucić wszystkie inserty.

Kolejny problem wystąpił kiedy podaliśmy datę w złym formacie. Wykonaliśmy zapytanie z datą o złym formacie:

```
INSERT INTO Book VALUES (1900,'Pan Tadeusz','fajna książka','10-01-1001');
```

Prawidłowy format to RRRR-MM-DD.

<div><div><div>↩</div><div>T</div><div>→</div></div><div>▼</div></div>						id	title	description	release_date
<div><div><div><div></div></div></div><div><div><div>✎</div></div>Edytuj</div><div><div><div>📄</div></div>Kopiuuj</div><div><div><div>🗑</div></div>Usuń</div></div>	1900	Pan Tadeusz	fajna książka	0000-00-00					

Rysunek 12. Prezentacja błędu związanego z datą

Jak widać na Rysunek 12 otrzymaliśmy zwrot 0000-00-00 co nie jest wyżej podaną datą.

## 5. Implementacja i testy aplikacji

### 5.1 Instalacja i konfigurowanie systemu



#### 5.1.1 Instalacja MySQL

Aby aplikacja zaczęła działać, należy zainstalować bazę danych MySQL. Zalecane programy ich wersje to ( *Rysunek 13* ):

	Product	Architecture	Installed	Upgrade To
<input checked="" type="checkbox"/>	 MySQL Server	X64	8.0.13	8.0.22
<input checked="" type="checkbox"/>	 MySQL Workbench	X64	8.0.13	8.0.22

*Rysunek 13. Zalecane wersje aplikacji*

W momencie tworzenia dokumentacji instalator można znaleźć pod adresem <https://dev.mysql.com/downloads/installer/> . Należy wybrać wersję instalatora 8.0.23. Po zakończeniu instalacji wybranych produktów, należy uruchomić instalator ponownie i wcisnąć przycisk *Cancel* .

Product	Version	Architecture	Quick Action
MySQL Server	8.0.13 	X64	<a href="#">Reconfigure</a>
MySQL Workbench	8.0.13 	X64	

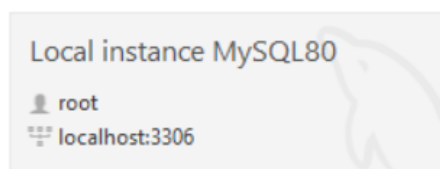
*Rysunek 14. Konfiguracja MySQL Server*

Ostatnim krokiem instalacji jest wybranie *Reconfigure* gdzie tworzymy użytkownika o loginie *root* i hasło *admin* .

#### 5.1.2 Konfiguracja Bazy danych

Uruchamiamy MySQL Workbench i używamy tego połączenia:

#### MySQL Connections



*Rysunek 15. Połączenia MySQL*

Wykonujemy zapytanie *create database biblioteka;* , następnie wykonujemy wszystkie kody *.sql* załączone razem z dokumentacją.

Aby dodać konto pracownika należy zarejestrować się według punktu **5.2.2.2**. Następnie odnaleźć dodanego użytkownika w bazie danych i zmienić jego **role** na `id == 1`.

```
UPDATE _User SET role_id = 1 WHERE login = "twoj login" AND password = "twoje hasło";
```

Rysunek 16. Przykładowa zmiana uprawnień usera na podstawie loginu i hasła

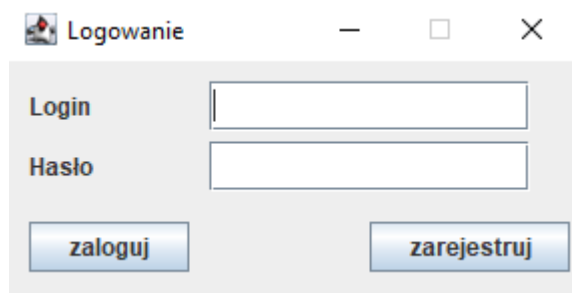
### 5.1.3 Uruchomienie aplikacji

Aby uruchomić aplikację trzeba mieć zainstalowaną *Java 1.8* ( jest to minimalna wersja, mogą być wyższą ). W folderze z plikiem *biblioteka.jar* (załączony razem z dokumentacją) uruchamiamy cmd i wpisujemy polecenie `java -jar biblioteka.jar`. Po tych krokach aplikacja jest gotowa do działania.

## 5.2 Instrukcja użytkownika aplikacji

### 5.2.1 Instrukcja Pracownika

#### 5.2.1.1 Logowanie



Rysunek 17. Okno logowania

Po uruchomieniu aplikacji pojawia się okno logowania. Okno jest tożsame zarówno dla pracownika jak i klienta.

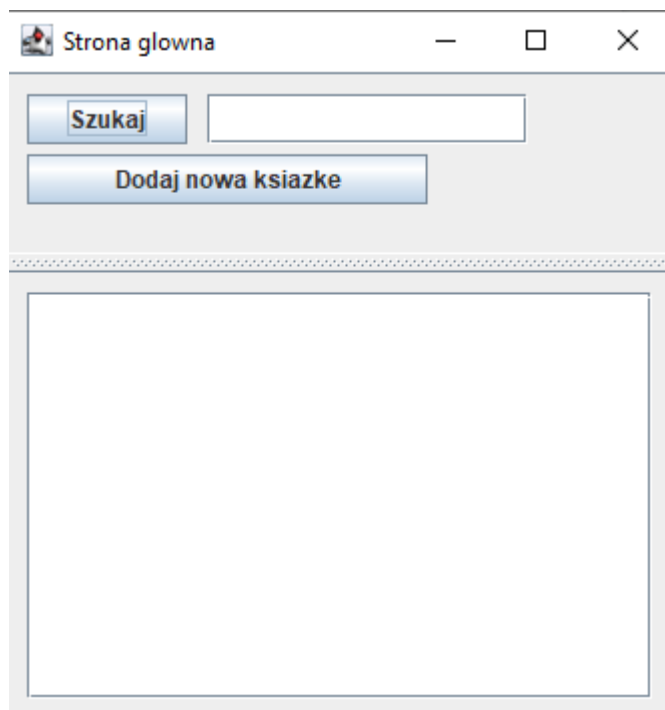
W celu zalogowania się należy wprowadzić prawidłowe dane logowania, następnie wcisnąć przycisk zaloguj.



Rysunek 18. Okno logowania wypełnione przykładowymi danymi logowania

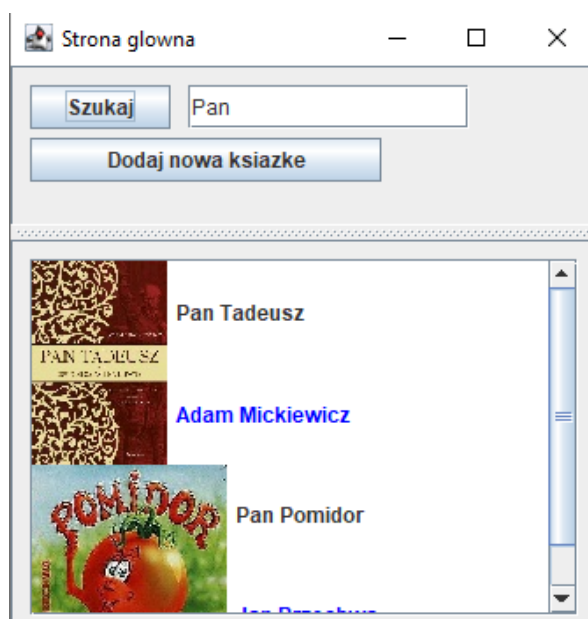
### 5.2.1.2 Przeglądanie książek

Po zalogowaniu klient powinien zobaczyć następujące okno. Służy ono do przeglądania książek dostępnych w bibliotece.



Rysunek 19. Okno główne przeglądu książek biblioteki.

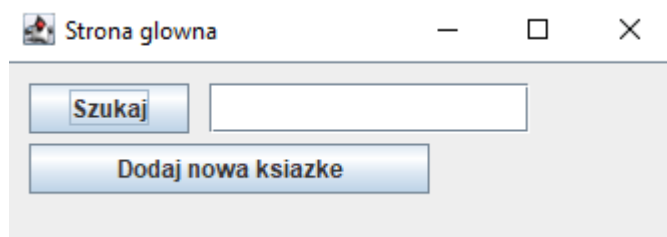
Po wpisaniu w pasek wyszukiwania dowolnej frazy, zostaną zwrócone wszystkie książki o tytule rozpoczynającym się wypisanymi słowami:



Rysunek 20. Okno główne wyboru książek po wpisaniu frazy "Pan" w pasek wyszukiwania.

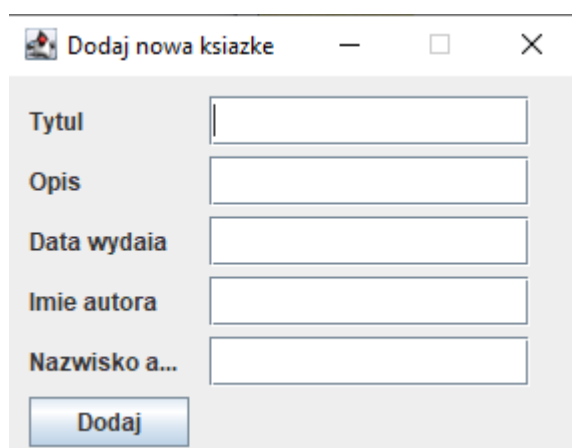
Nie wpisanie w pasek wyszukiwania żadnej frazy powoduje wyświetlenie wszystkich książek.

### 5.2.1.3 Dodawanie książek



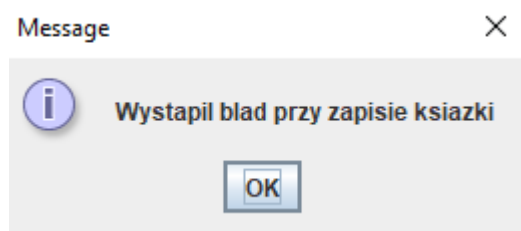
Rysunek 21. Okno główne wyboru książek - pasek wyszukiwania książki i przycisk

W celu dodania książki przez pracownika należy wcisnąć przycisk dodaj nową książkę. Powinno wyświetlić się wtedy następujące okno:



Rysunek 22. Okno "Dodaj nową książkę"

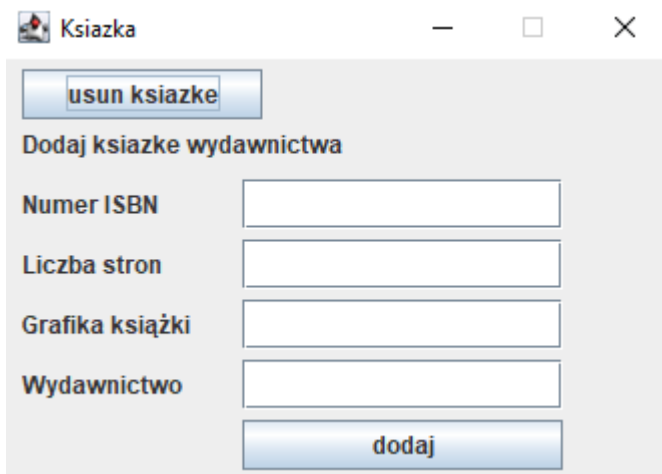
Po wypełnieniu pól poprawnymi danymi do biblioteki zostanie dodana nowa książka. W przypadku wypełnienia danych błędnymi wartościami zostanie zwrócony błąd:



Rysunek 23. Komunikat błędu

### 5.2.1.4 Dodawanie egzemplarzy

Dwukrotne kliknięcie książki w panelu wyszukiwania powoduje otwarcie edytora egzemplarzy.



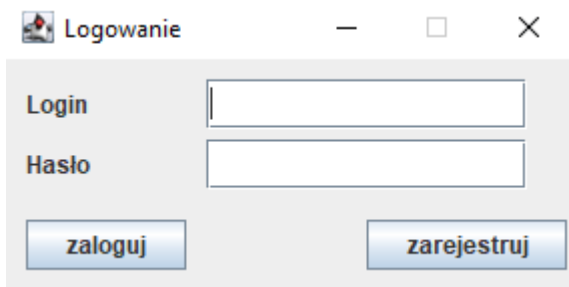
The screenshot shows a window titled 'Książka' with standard window controls (minimize, maximize, close). Inside the window, there is a button labeled 'usun książkę' at the top left. Below it, the section 'Dodaj książkę wydawnictwa' contains four input fields with labels: 'Numer ISBN', 'Liczba stron', 'Grafika książki', and 'Wydawnictwo'. At the bottom of this section is a button labeled 'dodaj'.

Rysunek 24. Okno edycji nowego egzemplarza książki

Edytor pozwala na dodanie nowych egzemplarzy danej książki do biblioteki. Pozwala także na usunięcie książki.

## 5.2.2 Instrukcja Klienta

### 5.2.2.1 Logowanie

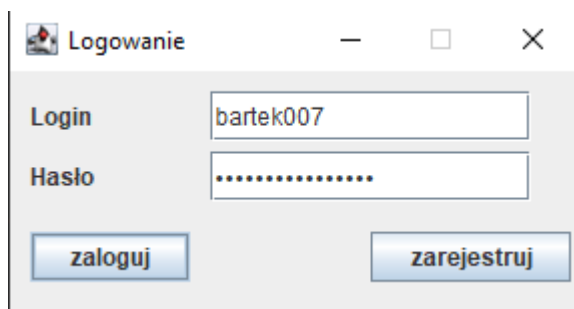


The screenshot shows a window titled 'Logowanie' with standard window controls (minimize, maximize, close). Inside the window, there are two input fields labeled 'Login' and 'Hasło'. Below these fields are two buttons: 'zaloguj' on the left and 'zarejestruj' on the right.

Rysunek 25. Okno logowania

Po uruchomieniu aplikacji pojawia się okno logowania. Okno jest tożsame zarówno dla pracownika jak i klienta.

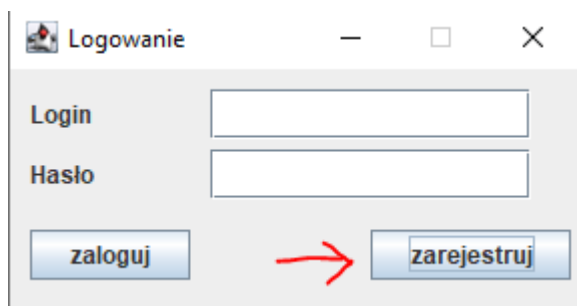
W celu zalogowania się należy wprowadzić prawidłowe dane logowania, następnie wcisnąć przycisk zaloguj.



Rysunek 26. Okno logowania z wypełnionymi danymi

### 5.2.2.2 Rejestracja

W przypadku braku konta zalecana jest rejestracja. W tym celu należy wcisnąć przycisk “zarejestruj”.



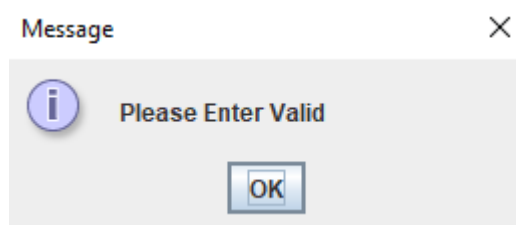
Rysunek 27. Okno logowania - przycisk rejestracji

W wyniku akcji powinno otworzyć się nowe okno z polami do wypełnienia. Pole należy wypełnić odpowiednimi danymi. Następnie należy wcisnąć przycisk “zarejestruj”.

The image shows a standard Windows-style window titled "Rejestracja" (Registration). It contains a vertical list of text input fields for the following labels: "imie" (first name), "nazwisko" (last name), "data urodz." (date of birth), "nick", "login", "haslo" (password), "email", "telefon" (phone), "miasto" (city), "ulica" (street), "numer domu" (house number), and "numer miesz." (apartment number). At the bottom of the form is a blue button labeled "Zarejestruj" (Register).

Rysunek 28. Okno rejestracji

Jeżeli wszystkie dane zostały wprowadzone poprawnie powinno zostać utworzone nowe konto. W przeciwnym przypadku zostanie zwrócony błąd:

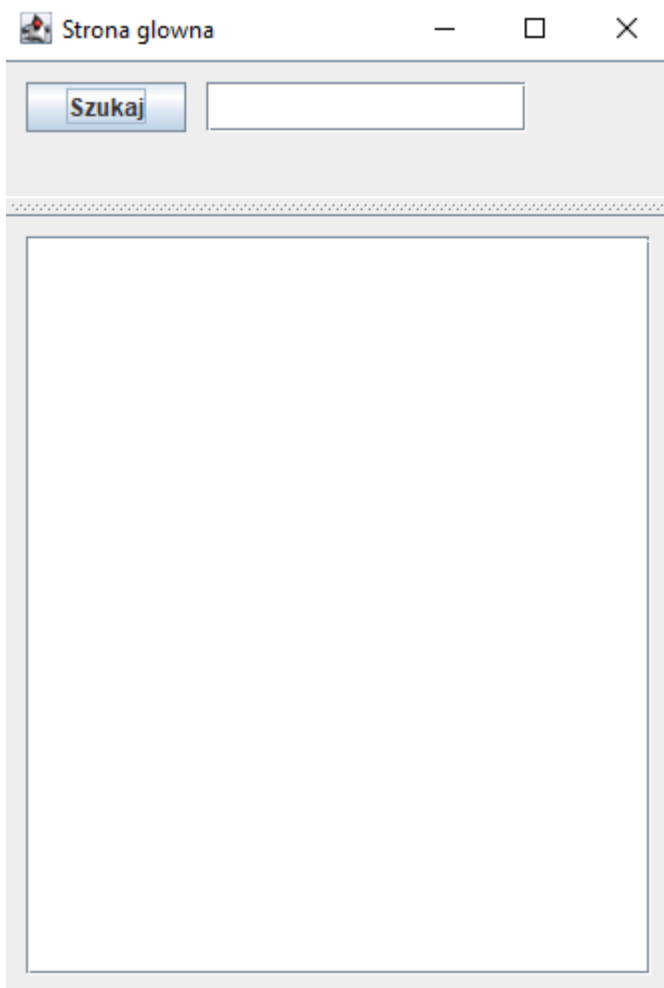


Rysunek 29. Wiadomość o błędzie danych wejściowych

### 5.2.2.3 Wyszukiwanie książek

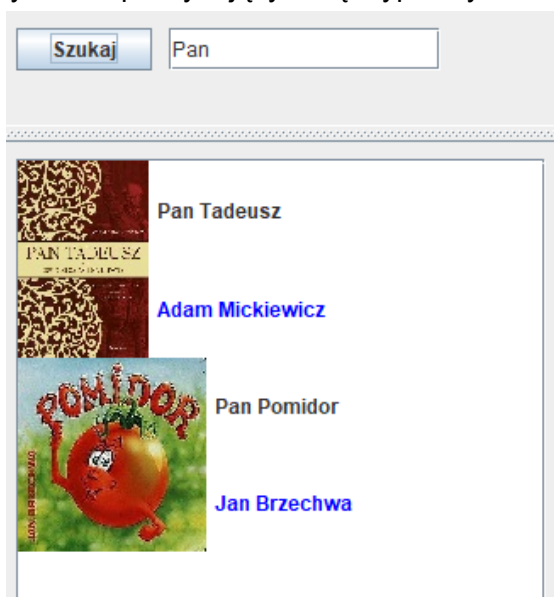
Po zalogowaniu klient powinien zobaczyć następujące okno. Służy ono do przeglądania książek dostępnych w bibliotece.





Rysunek 30. Okno strony głównej

Po wpisaniu w pasek wyszukiwania dowolnej frazy, zostaną zwrócone wszystkie książki o tytule rozpoczynającym się wypisanymi słowami:

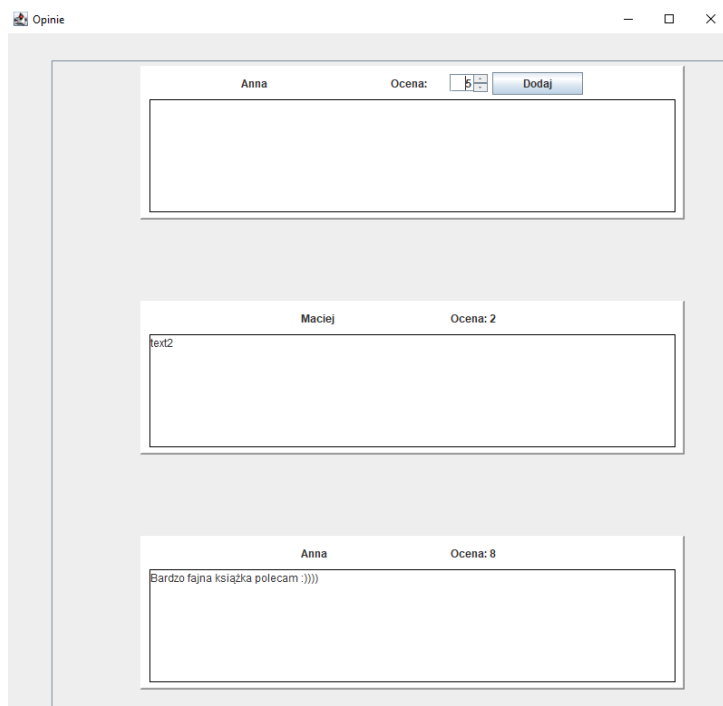


Rysunek 31. Okno wyszukiwania książki

Nie wpisanie w pasek wyszukiwania żadnej frazy powoduje wyświetlenie wszystkich książek.

#### 5.2.2.4 Opinie

Aby otworzyć okno opinii, należy nacisnąć na wyszukaną książkę. Powinien wtedy ukazać się następujący widok:



Rysunek 32. Okno przeglądu opinii

W nowym oknie ukazują się opinie pozostałych użytkowników. Na szczycie strony znajduje się pusta opinia. Można wypełnić ją własnym tekstem oraz oceną i samodzielnie dokonać oceny książki. Nie jest możliwe wystawienie 2 opinii, w wyniku próby wystawienia drugiej opinii, zostanie nadpisany tekst pierwszej.

### 5.3 Testowanie opracowanych funkcji systemu

#### 5.3.1 Testy jednostkowe

Poprawność implementacji oraz działania metod systemu sprawdzano przy pomocy testów jednostkowych JUnit. Wszystkie testy jednostkowe zostały zawarte w pakiecie test.java i zostały podzielone na odpowiednie pomniejsze pakiety zależnie od testowanych klas aplikacji.

```

@Test
public void shouldSaveUser() throws SQLException {
    //given
    User user = new User(null, "Kuba", "Kubakowski", "2000-10-10", 6, 6, 1, 1);
    //when
    userRepository.save(user);
    //then
    User actual = userRepository.findByNick("kubcio");
    user.id=actual.id;
    assertEquals(user.toString(), actual.toString());
    actual = userRepository.findByLogin("kubcio");
    assertEquals(user.toString(), actual.toString());
}

```

Rysunek 33. Przykładowe testy jednostkowe

Wszystkie testy jednostkowe zostały wykonane pomyślnie:

✓ Tests passed: 35 of 35 tests – 4 s 516 ms

Rysunek 34. Komunikat poprawności wykonania wszystkich testów jednostkowych

## 5.4 Omówienie wybranych rozwiązań programistycznych

5.4.1 Struktura kodu i implementacja wybranych funkcjonalności systemu.

Kod źródłowy został podzielony na pakiety na podstawie funkcjonalności. Hierarchia pakietów prezentuje się w następujący sposób:

- ▼ main
  - ▼ java
    - > controller
    - > exception
    - > factory
    - mapper
    - > model
    - > repository
    - > service
    - > ui
    - Biblioteka
    - > resources
  - > test

Rysunek 35. Struktura projektu "Biblioteka Wiejska"

- Images  
Images zawiera w sobie grafiki książek biblioteki
- Controller  
Controller jest klasą opakowującą wszystkie klasy typu service. Zwiększa ona wygodę użytkownika oraz czytelność kodu, gdyż wystarczy stworzyć jedną instancję klasy controller, aby otrzymać dostęp do wszystkich metod serwisów. W przeciwnym razie wymagane byłoby utworzyć instancję każdego serwisu z osobna.

```
public class Controller {

    public static Controller instance = new Controller();

    Statement statement = LocalStatementBuilder.getInstance().createStatement();

    private AuthService authService;
    private BookService bookService;
    private BookRepository bookRepository;

    public Controller() {
        createServices();
    }

    private void createServices(){
        bookRepository = new BookRepository(statement);
        AuthorRepository authorRepository = new AuthorRepository(statement);
        PublisherBookRepository publisherBookRepository = new PublisherBookRepository(statement);
        ContactRepository contactRepository = new ContactRepository(statement);
        AddressRepository addressRepository = new AddressRepository(statement);
        BookCopyRepository bookCopyRepository=new BookCopyRepository(statement);
        HireRepository hireRepository = new HireRepository(statement);
        StateRepository stateRepository=new StateRepository(statement);
        ReservationRepository reservationRepository = new ReservationRepository(statement);
        OpinionRepository opinionRepository=new OpinionRepository(statement);
        BookTypeRepository bookTypeRepository=new BookTypeRepository(statement);
        authService = new AuthService(new UserRepository(statement),new UserAuthenticationRepository(statement)
            ,contactRepository,addressRepository);
        bookService = new BookService(bookRepository,authorRepository,publisherBookRepository
            ,bookCopyRepository,hireRepository,stateRepository,reservationRepository
            ,opinionRepository,bookTypeRepository);
    }
}
```

Rysunek 36. Fragment kodu klasy Controller

- Exception  
Exception zawiera klasy zwracanych wyjątków.

```
package main.java.exception;

public class AuthException extends RuntimeException{
    public AuthException(String message) {
        super(message);
    }
}
```

Rysunek 37. Kod przykładowego wyjątku AuthException

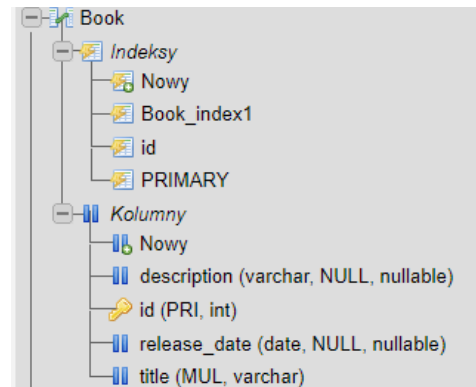
- Model  
Zawiera klasy-kontenery, umożliwiające reprezentacje strumienia danych otrzymanego z bazy danych sql w postaci obiektu zaimplementowanego w języku java.

```
package main.java.model;
```

```
public class Book {
```

```
    public int id;
    public String title;
    public String description;
    public String releaseDate;
```

```
    @Override
    public String toString() {
        return "Book{" +
            "id=" + id +
            ", title='" + title + '\'' +
            ", description='" + description + '\'' +
            ", releaseDate='" + releaseDate + '\'' +
            '}';
    }
}
```



Rysunek 38. Struktura książki Book w kodzie java (po lewej) oraz w bazie danych sql (po prawej)

- Repository

Repository zawiera klasy odpowiedzialne za wywoływanie metod na bazie danych. Dla każdej klasy modelu została utworzona odpowiadająca jej klasa repozytorium. W klasach tych zawarte są metody tłumaczące odpowiednie funkcje javy na zapytania w języku SQL.

```
public class AddressRepository extends Repository{

    public AddressRepository(Statement statement) {
        super(statement);
    }

    public void save(Address address) throws SQLException {
        String sql = "INSERT INTO Address(town,street,_number,house_number) VALUES(" +
            "\"" + address.town + "\", " +
            "\"" + address.street + "\", " +
            "\"" + address.houseNumber + "\"";
        StringBuilder sb = new StringBuilder(sql);
        if(address.street != null)
            sb.append(" + address.street + "\",");
        else
            sb.append("NULL,");
        sb.append(address.number + ",");
        sb.append(" + address.houseNumber + "\"");
        sql = sb.toString();
        statement.execute(sql);
    }
}
```

Rysunek 39. Fragment kodu przykładowej klasy typu Repository - AdressRepository

- Service

Service zawiera klasy służące do komunikacji między elementami UI oraz Repository. Wykorzystanie service pozwala ograniczyć redundancję kodów. Pobierają one dane z repozytoriów i zwracają w przystępnej dla wyższych warstw aplikacji postaci.

```

public List<BookDTO> getBooks(String title) throws SQLException {
    List<BookDTO> bookDTOS = new ArrayList<>();
    List<Book> books = bookRepository.selectThatBeginWith(title, "");

    for(Book b:books){
        BookDTO bookDTO=new BookDTO();
        Author a = new Author();
        a.name = "nieznany";
        a.lastName = "";
        if(authorRepository.findById(b.id).size()>0)
            a=authorRepository.findById(b.id).get(0);
        PublisherBook pb = new PublisherBook();
        pb.img = "images/notFound.jpg";
        if(publisherBookRepository.findById(b.id).size()>0)
            pb = publisherBookRepository.findById(b.id).get(0);
        bookDTO.title=b.title;
        bookDTO.description=b.description;
        bookDTO.img=pb.img;
        bookDTO.authorName=a.name;
        bookDTO.authorLastName=a.lastName;
        bookDTOS.add(bookDTO);
    }
    return bookDTOS;
}

```

Rysunek 40. Fragment przykładowej klasy typu Service - BookService

- UI  
UI zawiera klasy odpowiedzialne za warstwę ui aplikacji. Elementy graficzne zostały utworzone przy pomocy Java Swing.
- Test.java  
Zawiera klasy testów jednostkowych.

```

@Test
public void shouldSaveAddress() throws SQLException {
    //given
    Address address = new Address();
    address.town = "Garmish-Partenkirchen";
    address.street = "GoldenStrasse";
    address.number = 17;
    address.houseNumber = "34";
    //when
    addressRepository.save(address);
    //then
    Address returnedAddress = addressRepository.findByValues(address).get(0);
    address.id = returnedAddress.id;
    assertEquals(returnedAddress.toString(), address.toString());
}

```

Rysunek 41. Przykładowy kod testu jednostkowego

## 5.4.2 Implementacja interfejsu dostępu do bazy danych

Baza danych została wykonana w formie bibliotecznej aplikacji desktopowej. Łączy się ona z bazą danych MySQL za pomocą biblioteki `java.sql.connection`:

```
public class LocalStatementBuilder implements StatementBuilder{

    private static LocalStatementBuilder instance = null;

    private String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    private String DB_URL = "jdbc:mysql://localhost:3306/biblioteka?serverTimezone=UTC";

    private String USER = "root";
    private String PASS = "admin";

    private Connection connection;

    public LocalStatementBuilder() {

        try {
            Class.forName(JDBC_DRIVER);
            connection = DriverManager.getConnection(DB_URL, USER, PASS);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Rysunek 42. Kod dostępu do bazy danych SQL

Dane z bazy MySQL zostają zapisane w plikach `.sql`, z których następnie są odczytywane przez program:

```
public void resetDatabase() throws IOException, SQLException {
    ScriptRunner runner=new ScriptRunner(connection, false, false);
    InputStreamReader reader1 = new InputStreamReader(new FileInputStream("./src/main/resources/delete.sql"));
    InputStreamReader reader2 = new InputStreamReader(new FileInputStream("./src/main/resources/table.sql"));
    InputStreamReader reader3 = new InputStreamReader(new FileInputStream("./src/main/resources/view.sql"));
    InputStreamReader reader4 = new InputStreamReader(new FileInputStream("./src/main/resources/index.sql"));
    InputStreamReader reader5 = new InputStreamReader(new FileInputStream("./src/main/resources/example.sql"));
    runner.runScript(reader1);
    runner.runScript(reader2);
    runner.runScript(reader3);
    runner.runScript(reader4);
    runner.runScript(reader5);
    reader1.close();
    reader2.close();
    reader3.close();
    reader4.close();
    reader5.close();
}
```

Rysunek 43. Kod metody `resetDatabase` klasy `LocalStatementBuilder`

## 5.4.3 Implementacja mechanizmów bezpieczeństwa

### 5.4.3.1 System obsługi wyjątków

Aby uniknąć utraty kontroli nad poprawnym zachowaniem aplikacji w przypadku wystąpienia błędów, zastosowano system obsługi wyjątków.

```
loginButton.addActionListener(e -> {
    try {
        User user = cc.login(userText.getText(), passwordText.getText());
        frame.dispose();
        new MainPageView(user);
    } catch (SQLException e1) {
        JOptionPane.showMessageDialog(frame, "Niepoprawny login!");
    } catch (AuthException e2) {
        JOptionPane.showMessageDialog(frame, e2.getMessage());
    }
});
```

Rysunek 44. Przykładowe zastosowanie obsługi wyjątków

Głównym zadaniem wyjątków jest informowanie o tym, czy aplikacja poprawnie łączy się i wykonuje operacje na bazie danych. W przypadku niemożliwości zapisu lub odczytu danych z bazy zostanie zwrócony wyjątek, bez którego dojść mogłoby do krytycznych błędów.

#### 5.4.3.2 Umieszczenie aplikacji

Aplikacja działa na komputerze w bibliotece, co samo w sobie jest mechanizmem bezpieczeństwa. Komputery w bibliotece mają zainstalowane oprogramowanie uniemożliwiające na wiele sposobów potencjalnemu przestępcy zniszczenie, bądź kradzież danych.

## 6. Podsumowanie i wnioski

Etap ustalania wstępnych założeń projektowych oraz modelowania przebiegł bez żadnych problemów. Podobnie było z utworzeniem bazy danych MySQL. W celu ułatwienia sobie pracy wykupiliśmy dostęp do internetowego serwera gdzie przechowywana była nasza baza. Zrealizowaliśmy wszystkie zaplanowane zapytania w języku MySQL.

Pierwsze problemy pojawiły się podczas implementacji aplikacji dostępowej. Skala wstępnych założeń projektu znacznie przerosła nasze możliwości ich wykonania. Dlatego w trakcie realizacji kolejnych funkcjonalności aplikacji musieliśmy zrezygnować z części oferowanych przez nią funkcji. Ograniczyliśmy się do niezbędnego minimum i odrzuciliśmy wszystkie rozwiązania nie związane bezpośrednio z biblioteką.

Nie ma możliwości komunikowania się z innymi użytkownikami biblioteki drogą wiadomości. Pozostała jednak możliwość udzielania opinii pod bibliotecznymi książkami wraz z ich oceną.



Wypożyczanie książek także nie jest dostępne bezpośrednio przez aplikację. Służy ona bardziej prezentacji oferowanych towarów potencjalnemu konsumentowi, a sama finalizacja wypożyczenia musi nastąpić bezpośrednio w bibliotece. Nie jest to duża strata bowiem na etapie projektowania nie planowaliśmy wysyłki książek użytkownikom, zatem dokonywanie wypożyczeń zdalnie nie wносиły wiele do projektu.

Warstwa graficzna aplikacji także nie była naszym priorytetem podczas realizacji projektu. Głównym celem było nabycie umiejętności tworzenia baz danych sql oraz aplikacji dostępowych. Z tego względu aplikacja posiada minimalistyczny wygląd. Biorąc jednak pod uwagę fakt, iż głównym odbiorcą naszej aplikacji jest mieszkaniec wsi w podeszłym wieku, brak zbędnych rozpraszaczy oraz ogólny minimalizm graficzny może okazać się atutem.

## 7. Literatura

<https://www.tutorialspoint.com/jdbc/jdbc-sample-code.htm>

<https://cachhoc.net/2014/04/25/java-swing-tuy-bien-jlist-jlist-custom-renderer/?lang=en>

<https://strefakodera.pl/bazy-danych/sql/wstawianie-danych-do-tabel-w-sql>

[https://www.w3schools.com/sql/sql\\_create\\_table.asp](https://www.w3schools.com/sql/sql_create_table.asp)

[https://www.w3schools.com/php/php\\_mysql\\_create\\_table.asp](https://www.w3schools.com/php/php_mysql_create_table.asp)

<https://paiza.io/projects/6kY0NSgi8e2TfCK5ld7SWw?language=mysql>

<https://www.youtube.com/watch?v=PKVvn5oP4p8>

<https://pma.serverproject.eu/index.php?server=21>