**UNIWERSYTET RZESZOWSKI  
Kolegium Nauk Przyrodniczych**

Kacper Popowicz

134963

Informatyka I rok

**Projekt: ”** **System zarządzania biblioteką”**

**Dokumentacja techniczna**

**Rzeszów, 2025r.**

# Opis projektu

Jest to aplikacja do zarządzania uniwersalną biblioteką. Aplikacja łączy się z bazą danych i pozwala na przeprowadzanie za pomocą niej operacji typu CRUD. Posiada oddzielne menu dla czytelnika i administratora (pracownika). Projekt został napisany za pomocą JavaFX oraz JDBC. Do interfejsów graficznych został użyty SceneBuilder.

## Cel Projektu

Projekt ma być intuicyjną aplikacj, umożliwiającą zarządzanie biblioteką. Ma wprowadzać dane do bazy postgreSQL, po przez przyjazdne Menu, które w zależności kto jest zalogowany (czytelnik, czy administraor) pokażę inne rodzaj menu, tak żeby zwykły czytelnik miał wgląd w to co wypożyczył i nie miał opcji CRUD na danych.

# Baza danych dla biblioteki

Baza danych składa się z 4 tabel:

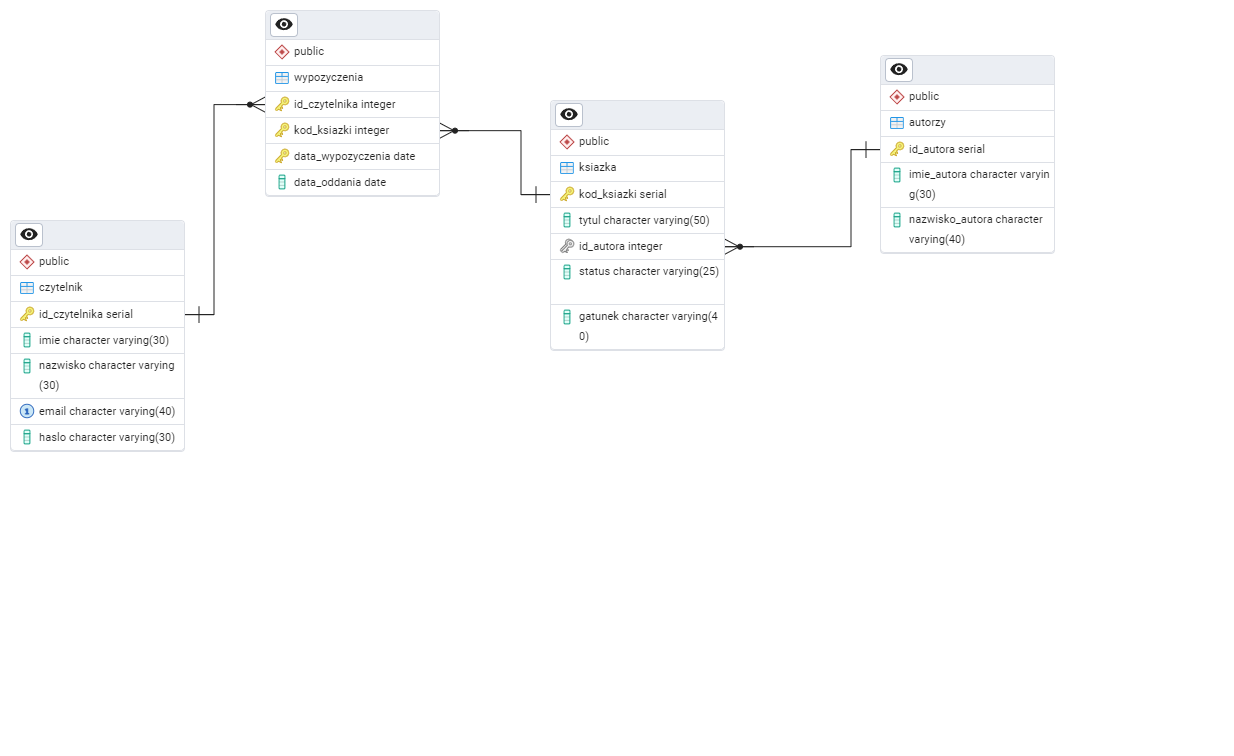
-Ksiazka,

-Autorzy,

-Wyporzyczenia,

-Czytelnik

## ERD Biblioteki



## Opis bazy danych Biblioteka2.0

Tabela Czytelnik:

Przechowóje dane zarejestrowanego czytelnika, jego imię, nazwisko, adres e-mail, oraz hasło.

Tabela Wypozyczenia:

Przechowóje dane na temat wyporzyczonej książki, mianowicie jej termin wyporzyczenia oraz oddania. Zawiera kod\_ksiazki oraz id\_czytelnika, dzięki czemu będziemy mogli zobczyć kto wyporzyczył jaką książkę.

Tabela Książka:

Zawiera informacje o tytule książki jej gatunku i statusie(wyporzyczona/dostępna). Dzięki kluczowi obcemu id\_autora daje nam możliwość podejżenia kto jest autorem daje ksiązki.

Tabela Autorzy:

Usytuowane w niej są informacje o imieniu i nazwisku danego autora.

Relacje między tabelami:

Tabele są połączone między sobą relacjami co pozwala nam na sprawne pozyskiwanie informacji w nich zawartych.

# Gui aplikacji

## Okno logowania się do aplikacji

Obraz zawierający tekst, zrzut ekranu, Czcionka, design

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Minimalistyczne okienko, pozwalające nam na zalogowanie się do aplikacji, w przypadku gdy dany czytelnik nie ma konta, może je zrobićklikając w przycisk „zarejestruj się”.

### Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.Widok w kodzie

Przyciski i pola mają odpowiednio przypisane id i funkcje.

Okno wyświetla też alert, gdy zostaną wpisane błędne dane:

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

## Okno rejestracji

Obraz zawierający tekst, zrzut ekranu, Czcionka, design

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Okno pomaga w rejestracji dzieki powiadomieniom na niej:

Obraz zawierający tekst, zrzut ekranu, Czcionka, design

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.Obraz zawierający tekst, zrzut ekranu, Czcionka, design

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

## Okno menu czytelnika

Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Pozwala na zobaczenie czytelnikowi, jakie książki ma aktualnie wypozyczone oraz informacje o nich wraz z datą wyporzyczenia takowej. Aby byćpewnym, że czytelnik widzi swoje książki dodałem informacje w lewym rogu u góry, jakie konto jest aktualnie zalogowane.

## Okno menu administratora (pracownika)

Obraz zawierający tekst, zrzut ekranu, paragon

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.Po zalogowaniu się na odpowiednie konto, ukazuje się nam oto takie menu, pokazujące zarejestrowanych czytelników:

Obraz zawierający zrzut ekranu, tekst, paragon

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.Dodatkowo mamy przycisk do usuwania czyelników, wystarczy kliknąć na czytelnika, którego chcemu usunąć i nacisnąć przycisk:

Obraz zawierający tekst, numer, Czcionka, zrzut ekranu

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.Gdy klikniemy w znajdujący się po lewej stronie przycisk „Książki”, wyświetli się tabela z książkami znajdującymi się w bibliotece:

A po klknięciu w „Wypożyczenia”, wyświetlą nam się aktualnie wypozyczone książki, wraz z informacją przez kogo i kiedy zostały wyporzyczone:

Obraz zawierający tekst, zrzut ekranu, numer

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Obraz zawierający tekst, zrzut ekranu

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.Mamy, tu też przycisk pozwalający nam na zmiane statusu książki, kiedy dany czytelnik odda nam książkę:

Wracając do panelu „Książki”, widzimy tam kilka ciekawych przycisków:

-„Usuń Książkę”

-„Dodaj książkę”

-„Wypożycz książkę”

„Usuń Książkę” – usuwa wybraną książkę.

„Wypożycz książkę” – po wpisaniu emaila osoby chcącej wypozyczyć daną książkę

Klikamy przycisk i wyporzyczamy ją daje osobie.

„Dodaj książkę” – Wyświtla nam okno, umożliwiające dodanie książki:

Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

## Controller

Każde GUI ma swój własny plik Controller, który definiuje, co dane elementy w GUI mają robić.

### Przykład

Plik LoginController:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

OnZalogujSie() – to funkcja przycisku „Zaloguj się”. Sprawdza czy w TextField (czyli w email i hasło), zostały wpisane dane, jeśli nie, wyświetla nam błąd w oknie pod hasłem. Jednak, gdy są poprawne, podaje je dalej do funkcji LoginSpr, która sprawdza je dokładniej i otwiera odpowiednie menu.

# Kod źródłowy

Podgląd na klasy:

Obraz zawierający tekst, zrzut ekranu

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

## Classy a tabele

Każda tabela z bazy ma własną klasę z set-erami i get-erami, np. classa „Ksiazka”:

*package* com.example.biblioteka;  
  
*import* javafx.beans.property.SimpleStringProperty;  
  
*public class* Ksiazka {  
 *private final int* kodKsiazki;  
 *private final* SimpleStringProperty tytul;  
 *private final* SimpleStringProperty status;  
 *private final* SimpleStringProperty gatunek;  
 *private final* SimpleStringProperty imieAutora;  
 *private final* SimpleStringProperty nazwiskoAutora;  
  
 *public* Ksiazka(*int* kodKsiazki, String tytul, String status, String gatunek, String imieAutora, String nazwiskoAutora) {  
 *this*.kodKsiazki = kodKsiazki;  
 *this*.tytul = *new* SimpleStringProperty(tytul);  
 *this*.status = *new* SimpleStringProperty(status);  
 *this*.gatunek = *new* SimpleStringProperty(gatunek);  
 *this*.imieAutora = *new* SimpleStringProperty(imieAutora);  
 *this*.nazwiskoAutora = *new* SimpleStringProperty(nazwiskoAutora);  
 }  
  
 *public int* getKodKsiazki() {  
 *return* kodKsiazki;  
 }  
  
 *public* String getTytul() {  
 *return* tytul.get();  
 }  
  
 *public* String getStatus() {  
 *return* status.get();  
 }  
  
 *public* String getGatunek() {  
 *return* gatunek.get();  
 }  
  
 *public* String getImieAutora() {  
 *return* imieAutora.get();  
 }  
  
 *public* String getNazwiskoAutora() {  
 *return* nazwiskoAutora.get();  
 }  
  
 *public void* setTytul(String tytul) {  
 *this*.tytul.set(tytul);  
 }  
  
 *public void* setStatus(String status) {  
 *this*.status.set(status);  
 }  
  
 *public void* setGatunek(String gatunek) {  
 *this*.gatunek.set(gatunek);  
 }  
  
 *public void* setImieAutora(String imieAutora) {  
 *this*.imieAutora.set(imieAutora);  
 }  
  
 *public void* setNazwiskoAutora(String nazwiskoAutora) {  
 *this*.nazwiskoAutora.set(nazwiskoAutora);  
 }  
  
  
}

## Usuwanie danych z tabeli

Do usuwania danych z tabeli zrobiłem specjalną klasę posiadającą dwie odpowiedzialne za to klasy:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, System operacyjny

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Funkcja „usunKsiazkeZBazy” do usuwania książki, oraz

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Funkcję usunCzytelnikaZBazy do usuwania czytelnika z bazy.

Obie klasy używają funkcji JDBC do łączenia się z bazą:



Obraz zawierający tekst, zrzut ekranu, Czcionka

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.Używają też podobnych funkcji napisanych w jezyku SQL:

## Dodawanie danych do bazy

Do dodawania danych do Bazy jest kilka klasy w zależności od tego co chcemy dodać, o to przykładowa klasa „KsiazkaDodaj”, która pozwala nam na dodanie książki do bazy.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

## Aktualizowanie danych w bazie

Aplikacja zawiera również możliwość edycji, wcześniej wprowadzonych danych, przykładem może być klasa „aktualizacjaKsiazkiBaza”.

*package* com.example.biblioteka;  
  
*import* java.sql.*Connection*;  
*import* java.sql.DriverManager;  
*import* java.sql.*PreparedStatement*;  
*import* java.sql.SQLException;  
  
*public class* aktualizacjaKsiazkiBaza {  
  
 *public boolean* aktualizujKsiazkeWBazie(Ksiazka ksiazka) *throws* SQLException {  
 *Connection* conn;  
 String sql = """  
 UPDATE autorzy  
 SET imie\_autora = ?, nazwisko\_autora = ?  
 WHERE id\_autora = (  
 SELECT id\_autora FROM ksiazka WHERE kod\_ksiazki = ?  
 );  
  
 UPDATE ksiazka  
 SET tytul = ?, status = ?, gatunek = ?  
 WHERE kod\_ksiazki = ?  
 """;  
  
 *try* {  
 conn = DriverManager.*getConnection*("jdbc:postgresql://localhost:5432/Biblioteka2.0", "postgres", "kacper13");  
 *PreparedStatement* stmt = conn.prepareStatement(sql);  
  
 stmt.setString(1, ksiazka.getImieAutora());  
 stmt.setString(2, ksiazka.getNazwiskoAutora());  
  
 stmt.setInt(3, ksiazka.getKodKsiazki());  
  
 stmt.setString(4, ksiazka.getTytul());  
 stmt.setString(5, ksiazka.getStatus());  
 stmt.setString(6, ksiazka.getGatunek());  
  
 stmt.setInt(7, ksiazka.getKodKsiazki());  
  
 stmt.executeUpdate();  
 *return true*;  
 } *catch* (SQLException e) {  
 e.printStackTrace();  
 *return false*;  
 }  
 }  
}

## Wczytywanie danych z bazy

Najważniejszą rzeczą w aplikacjach do zarządzania danymi w bazie, są metody do sczytywania odpowiednich danych i ich wypisywanie, nie inaczej jest w tutaj, przykładową classą zawierającą tego typu funkcje jest np. class „ZczytanieKsiazek”.

*package* com.example.biblioteka;  
  
*import* javafx.collections.*ObservableList*;  
*import* javafx.scene.control.TableView;  
  
*import* java.sql.\*;  
  
*public class* ZczytanieKsiazek {  
  
 *public void* wczytajDaneZKsiazki(*ObservableList*<Ksiazka> listaKsiazek, TableView<Ksiazka> tabelaKsiazek, String szukanyCiag) {  
 *try* {  
 Class.*forName*("org.postgresql.Driver");  
 *Connection* conn = DriverManager.*getConnection*(  
 "jdbc:postgresql://localhost:5432/Biblioteka2.0",  
 "postgres",  
 "kacper13"  
 );  
  
 String zapytanie = "SELECT k.kod\_ksiazki, k.tytul, k.status, k.gatunek, a.imie\_autora, a.nazwisko\_autora " +  
 "FROM ksiazka k " +  
 "JOIN autorzy a ON k.id\_autora = a.id\_autora";  
  
 *PreparedStatement* stmt;  
 szukanyCiag = szukanyCiag == *null* ? "" : szukanyCiag.trim();  
  
 *if* (!szukanyCiag.isEmpty()) {  
 zapytanie += " WHERE LOWER(k.tytul) LIKE LOWER(?) " +  
 "OR LOWER(a.imie\_autora) LIKE LOWER(?) " +  
 "OR LOWER(a.nazwisko\_autora) LIKE LOWER(?)";  
  
 stmt = conn.prepareStatement(zapytanie);  
  
 String pattern = "%" + szukanyCiag + "%";  
  
 stmt.setString(1, pattern);  
 stmt.setString(2, pattern);  
 stmt.setString(3, pattern);  
  
 } *else* {  
 stmt = conn.prepareStatement(zapytanie);  
 }  
  
 *ResultSet* rs = stmt.executeQuery();  
  
 listaKsiazek.clear();  
  
 *while* (rs.next()) {  
 listaKsiazek.add(*new* Ksiazka(  
 rs.getInt("kod\_ksiazki"),  
 rs.getString("tytul"),  
 rs.getString("status"),  
 rs.getString("gatunek"),  
 rs.getString("imie\_autora"),  
 rs.getString("nazwisko\_autora")  
 ));  
 }  
  
 tabelaKsiazek.setItems(listaKsiazek);  
  
 } *catch* (SQLException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
}

## Klasy sprawdzające

W aplikacji znajdują się też klasy sprawdzające, które sprawdzają poprawność danych i na podstawie wytycznych, wykonują odpowiednie instrukcje w zależności od wyniku sprawdzenia.

*package* com.example.biblioteka;  
  
*import* java.sql.*Connection*;  
*import* java.sql.DriverManager;  
*import* java.sql.*PreparedStatement*;  
*import* java.sql.*ResultSet*;  
*import* javafx.fxml.FXML;  
*import* javafx.fxml.FXMLLoader;  
*import* javafx.scene.Scene;  
*import* javafx.scene.control.Label;  
*import* javafx.stage.Stage;  
  
  
*public class* LoginSprawdzanie{  
  
 @FXML  
 *protected* Label BladLogowania;  
  
 *public void* LoginSpr(String email, String haslo){  
  
 *Connection* conn;  
 *try* {  
 Class.*forName*("org.postgresql.Driver");  
 conn = DriverManager.*getConnection*("jdbc:postgresql://localhost:5432/Biblioteka2.0", "postgres", "kacper13");  
 *if* (conn != *null*) {  
 System.***out***.println("Połączono");  
  
 String zapytanie = "Select \* FROM czytelnik WHERE email = ? AND haslo = ?";  
  
 *PreparedStatement* preparedStatement = conn.prepareStatement(zapytanie);  
  
  
 preparedStatement.setString(1, email);  
 preparedStatement.setString(2, haslo);  
  
 *ResultSet* resultSet = preparedStatement.executeQuery();  
  
  
 *if* (resultSet.next()) {  
 *if* ( resultSet.getString("email").equals("admin@") && resultSet.getString("haslo").equals("admin123")){  
 System.***out***.println("Logowanie udane!");  
 FXMLLoader fxmlLoader2 = *new* FXMLLoader(Login.*class*.getResource("Menu.fxml"));  
 Scene scene2 = *new* Scene(fxmlLoader2.load(), 1200, 400);  
 Stage stage = *new* Stage();  
 stage.setTitle("Menu");  
 stage.setScene(scene2);  
 stage.show();  
 BladLogowania.getScene().getWindow().hide();  
 }  
 *else* {  
 System.***out***.println("Logowanie udane!");  
 String emailBazy = resultSet.getString("email");  
 NoweOkno okno = *new* NoweOkno();  
 okno.otworzOkno("MenuCzytelnika.fxml", "Menu", emailBazy);  
 BladLogowania.getScene().getWindow().hide();  
 }  
  
  
  
 } *else* {  
 BladLogowania.setText("Błędne dane!!!");  
  
 }  
  
  
 preparedStatement.close();  
 conn.close();  
  
  
  
  
 } *else* {  
 System.***out***.println("Nie połączono");  
 }  
 } *catch* (Exception e) {  
 System.***out***.println(e.getMessage());  
 }  
  
 }  
}

Funkcja ta łączy się z bazą danych i sprawdza, czy dane są prawidłowe (czyli czy znajdują się w bazie) oraz w zależności czy loguje się administrator (pracownik) lub czytelnik, otwiera przypisane im menu. Czyli dla użytkownika otworzy okrojone menu bez możliwości CRUD na danych a dla administratora, otworzy pełne menu z wszystkimi opcjami.

## Wnioski

Program działa sprawnie oraz zgodnie z poleceniem i założeniem, został on przetestowany pod względem błędnych danych. Funkcjonuje bez większych zarzutów. Ma minimalistyczne i proste GUI więc nowy użytkownik nie zagubi się w nim oraz nic w nim nie będzie go rozpraszać.

# Podsumowanie

Projekt aplikacji „System zarządzania biblioteką”, został zrealizowany pomyślnie. Został dogłębnie zanalizowany pod kątem danych wejściowych, danych wyjściowych, funkcji i przykładowych błędów. Został on wykonany w języku Java oraz z użyciem języka SQL. Aplikacja opiera się głównie o bibliotekę JavaFX oraz JDBC, Baza danych zostałą napisana w PostgreSQL. Program spełnia swoje zadanie, cel i założenia.