

Wypożyczalnia filmów

Autorzy:

Charkiewicz Kamil

Pogorzelski Artur

Prowadząca:

dr inż. Eugenia Busłowska

Spis treści

1. Wstęp	3
2. Analiza wymagań systemu.....	3
2.1 Wymagania funkcjonalne	3
2.2 Wymagania нефunkcjonalne.....	3
2.3 Diagram przypadków użycia	4
3. Wykorzystane technologie	4
4. Projekt i implementacja aplikacji.....	5
4.1 Architektura aplikacji.....	5
4.2 Projekt koncepcyjny bazy danych.....	6
4.3 Projekt schematu relacyjnego	7
5. Funkcjonalność aplikacji.....	7
6. Interfejs użytkownika.....	8
7. Podsumowanie	9
Dodatek A: Skrypty tworzące obiekty baz danych.....	9

1. Wstęp

Przedmiotem projektu jest aplikacja umożliwiająca zarządzanie wypożyczalnią filmów oraz korzystanie z jej usług.

2. Analiza wymagań systemu

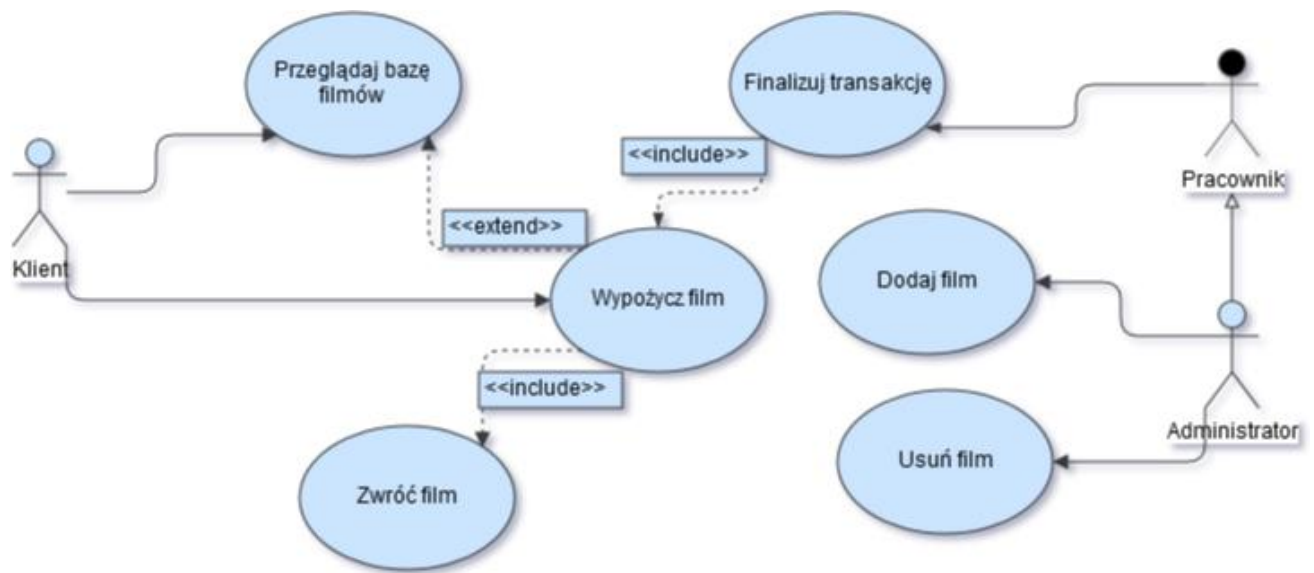
2.1 Wymagania funkcjonalne

- Umożliwienie klientowi przeglądania bazy filmów oraz sortowania jej,
- wyposażenie programu w funkcję pozwalającą na wypożyczenie filmu klientowi,
- panel administracyjny pozwalający administratorom na dodawanie nowych i usuwanie istniejących filmów w bazie,
- widok przedstawiający pracownikowi wypożyczalni transakcje oczekujące na weryfikację oraz pozwalający na ich finalizację,
- weryfikacja użytkowników programu przy użyciu logowania.

2.2 Wymagania niefunkcjonalne

- Przede wszystkim program powinien być intuicyjny i przyjazny użytkownikom,
- aplikacja powinna posiadać interfejs graficzny,
- aplikacja działająca na kilku platformach (Windows, Linux),
- aplikacja przeznaczona dla wielu użytkowników,
- aplikacja wymaga do działania Internetu,
- aplikacja powinna stawiać na wydajność pracy z bazą danych, mniej na bezpieczeństwo, ponieważ za zadanie ma prezentować możliwości współpracy z RDBMS,
- aplikacja łatwoskalowalna, z możliwością dalszej rozbudowy.

2.3 Diagram przypadków użycia

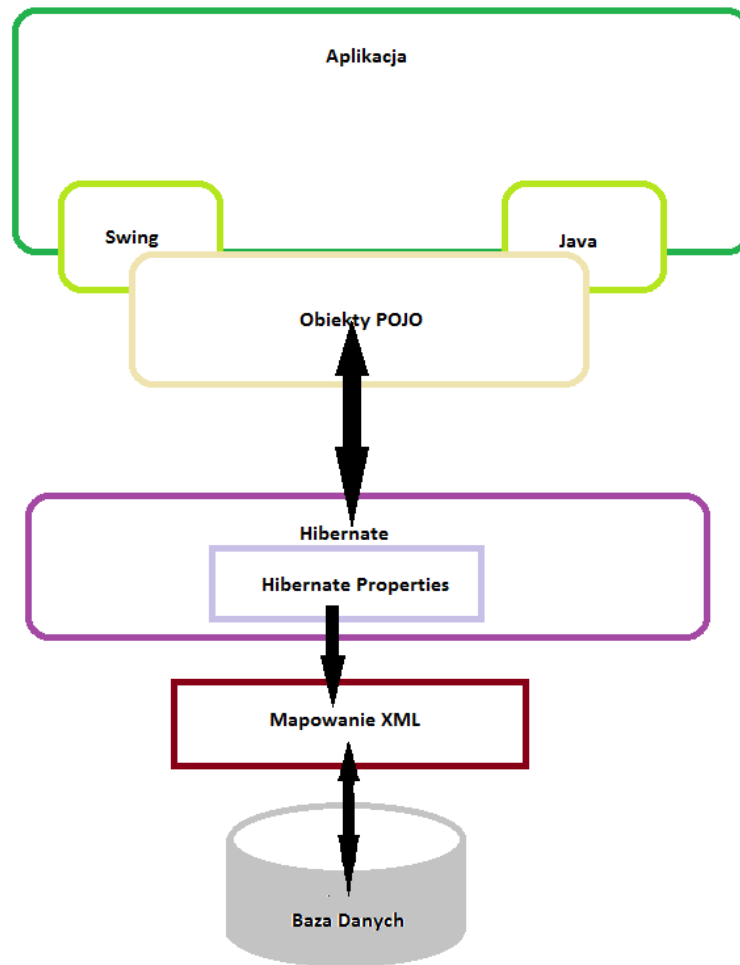


3. Wykorzystane technologie

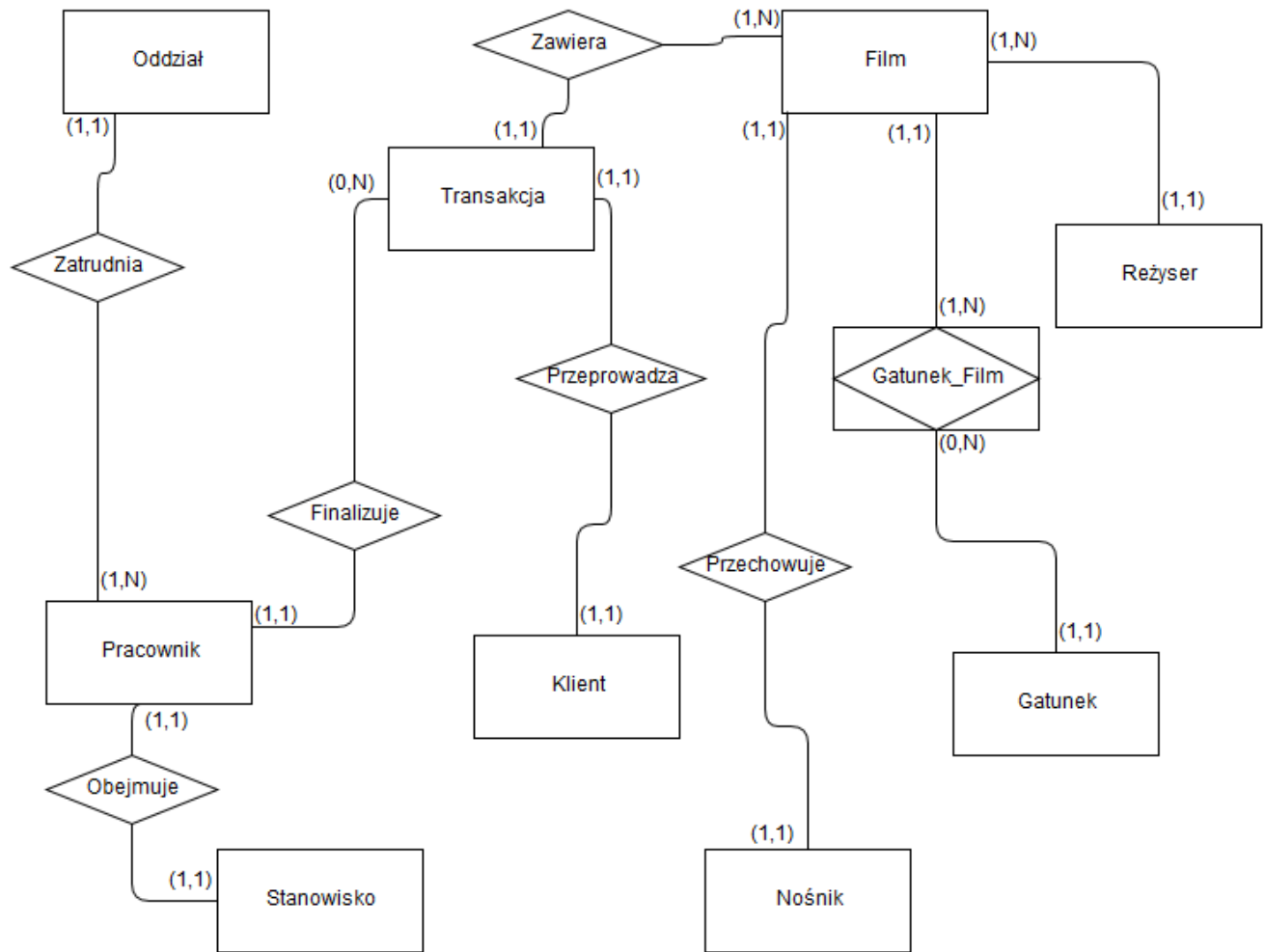
- Java,
- Hibernate,
- Koncepcja POJO,
- Swing,
- Apache Derby,
- Mapowanie obiektowo-relacyjne (ORM),

4. Projekt i implementacja aplikacji

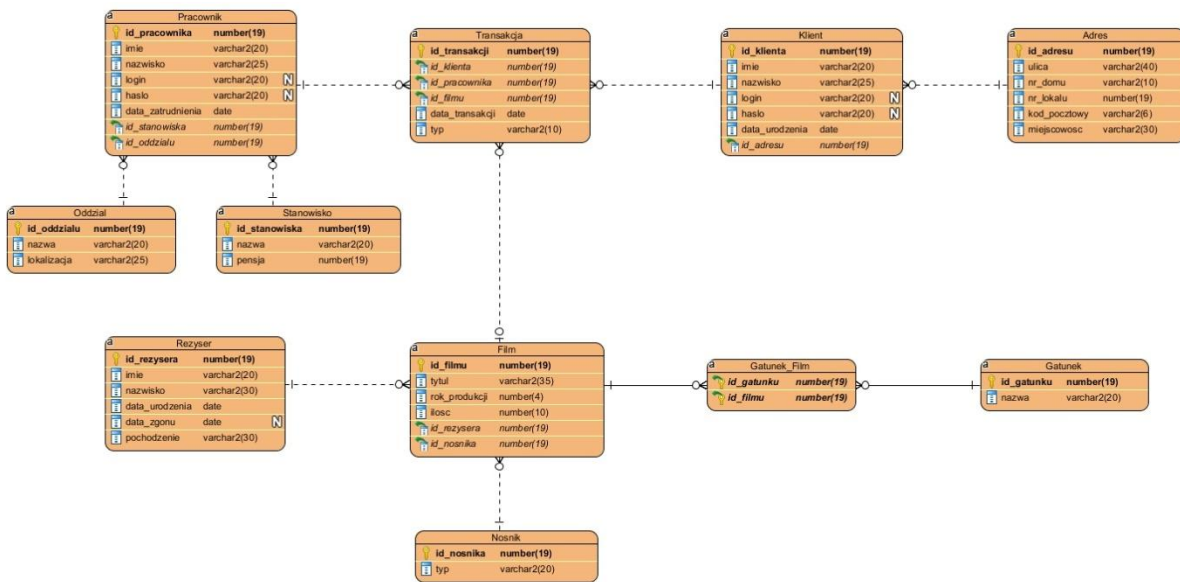
4.1 Architektura aplikacji



4.2 Projekt koncepcyjny bazy danych



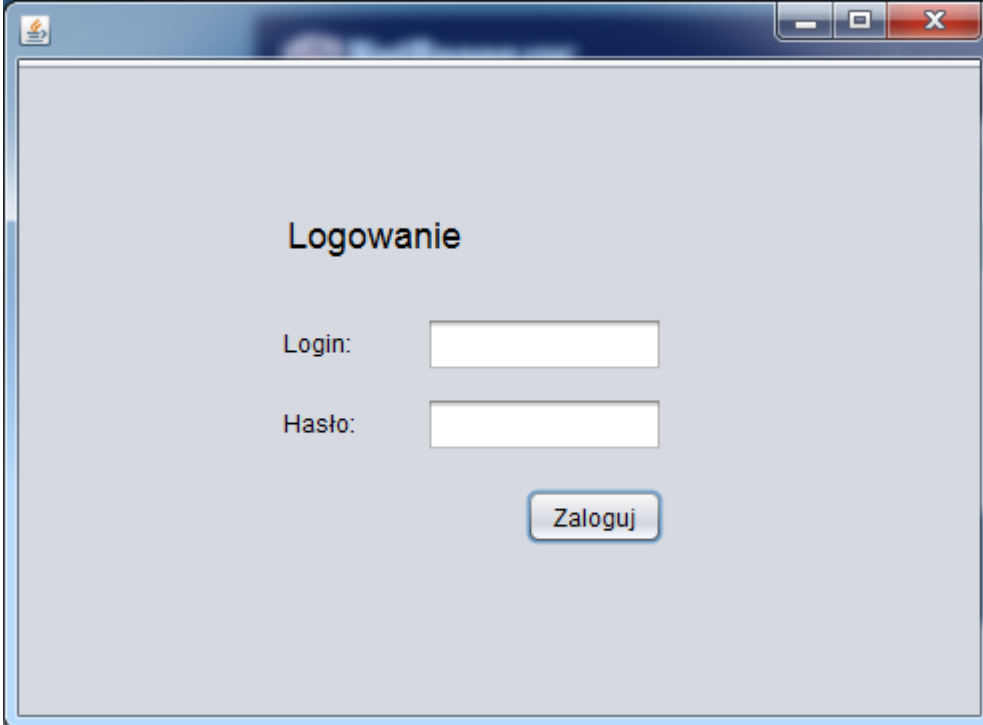
4.3 Projekt schematu relacyjnego



5. Funkcjonalność aplikacji

- Umożliwienie klientowi przeglądania bazy filmów oraz sortowania jej,
- wyposażenie programu w funkcję pozwalającą na wypożyczenie filmu klientowi,
- panel administracyjny pozwalający administratorom na dodawanie nowych i usuwanie istniejących filmów w bazie,
- widok przedstawiający pracownikowi wypożyczalni transakcje oczekujące na weryfikację oraz pozwalający na ich finalizację,
- weryfikacja użytkowników programu przy użyciu logowania.

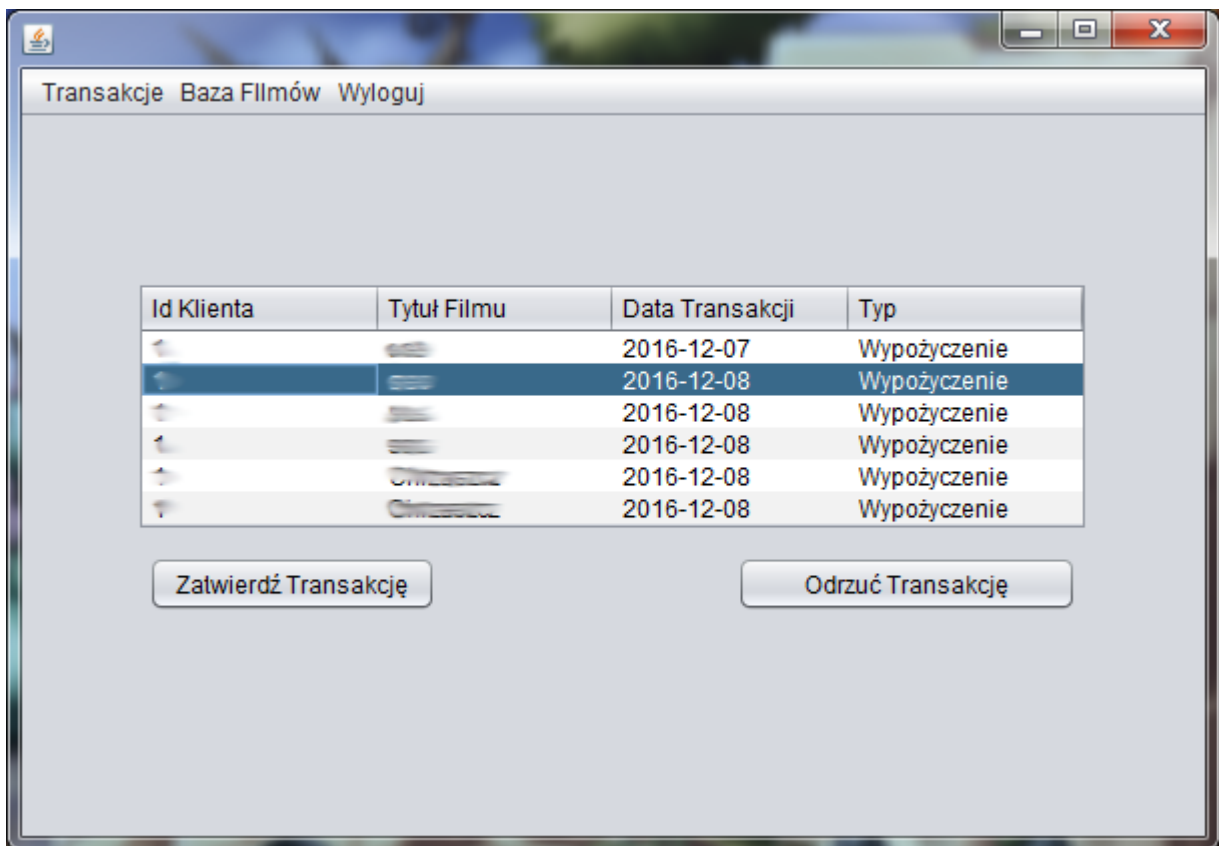
6. Interfejs użytkownika



The image shows a screenshot of a login window. The window has a title bar with a small icon on the left and standard Windows window controls (minimize, maximize, close) on the right. The main area of the window is light gray and contains the following elements:

- The title "Logowanie" is centered at the top in a bold, black font.
- Below the title, there are two labels: "Login:" and "Hasło:", each followed by a white rectangular text input field.
- Below the input fields, there is a button with the text "Zaloguj" in a blue font, set against a light gray background with a thin blue border.

Rys. 1. Ekran logowania



Rys. 2. Widok transakcji oczekujących na finalizację

7. Podsumowanie

Powstała aplikacja jest zgodna z założeniami, prosta i intuicyjna w użyciu, a także pozbawiona nadmiaru niepotrzebnych funkcjonalności mogących konfundować użytkowników.

Dodatek A: Skrypty tworzące obiekty baz danych

```
CREATE TABLE Gatunek
```

```
(
```

```
id_gatunku NUMERIC CONSTRAINT gatunek_pk PRIMARY KEY ,
```

```
nazwa VARCHAR(20)
```

```
);
```

```
CREATE TABLE Adres
```

```
(id_adresu NUMERIC CONSTRAINT adres_pk PRIMARY KEY ,
```

```
ulica VARCHAR(40),
```

```
nr_domu VARCHAR(10),
```

```
nr_lokalu NUMERIC,
```

```
kod_pocztowy VARCHAR(6),
```

```
miescowosc VARCHAR(30)
```

```
);
```

```
CREATE TABLE Stanowisko
```

```
(
```

```
id_stanowiska NUMERIC CONSTRAINT stanowisko_pk PRIMARY KEY ,
```

```
nazwa VARCHAR(20),
```

```
pensja NUMERIC CONSTRAINT pensja_ck CHECK (pensja>0)
```

```
);
```

```
CREATE TABLE Rezyser
```

```
(
```

```
id_rezysera NUMERIC CONSTRAINT rezyser_pk PRIMARY KEY ,
```

```
imie VARCHAR(20),
```

```
nazwisko VARCHAR(30) NOT NULL,
```

```
data_urodzenia DATE,
```

```
data_zgonu DATE,
```

```
pochodzenie VARCHAR(30)
```

```
);
```

```
CREATE TABLE Nosnik
```

```
(
```

```
id_nosnika NUMERIC CONSTRAINT nosnik_pk PRIMARY KEY ,
```

```
typ VARCHAR(20)
```

```
);
```

CREATE TABLE Film

```
(  
id_filmu NUMERIC CONSTRAINT film_pk PRIMARY KEY ,  
tytul VARCHAR(35) NOT NULL,  
rok_produkcji NUMERIC(4) CONSTRAINT rok_ck CHECK (rok_produkcji>=1895),  
ilosc NUMERIC(10) CONSTRAINT ilosc_ck CHECK (ilosc>=0),  
id_rezysera NUMERIC CONSTRAINT film_rezyser_fk REFERENCES Rezyser(id_rezysera),  
id_nosnika NUMERIC CONSTRAINT film_nosnik_fk REFERENCES Nosnik(id_nosnika)  
);
```

CREATE TABLE Oddzial

```
(  
id_oddzialu NUMERIC CONSTRAINT oddzial_pk PRIMARY KEY ,  
nazwa VARCHAR(20),  
lokalizacja VARCHAR(25)  
);
```

CREATE TABLE Pracownik

```
(  
id_pracownika NUMERIC CONSTRAINT pracownik_pk PRIMARY KEY ,  
imie VARCHAR(20),  
nazwisko VARCHAR(25),  
data_zatrudnienia DATE,  
id_stanowiska NUMERIC CONSTRAINT pracownik_stanowisko_fk REFERENCES Stanowisko(id_stanowiska),  
id_oddzialu NUMERIC CONSTRAINT pracownik_oddzial_fk REFERENCES Oddzial(id_oddzialu)  
);
```

CREATE TABLE Gatunek_Film

```
(  
id_gatunku NUMERIC CONSTRAINT gatunek_fk REFERENCES Gatunek(id_gatunku),  
id_filmu NUMERIC CONSTRAINT film_fk REFERENCES Film(id_filmu),  
CONSTRAINT gatunek_film_pk PRIMARY KEY (id_gatunku,id_filmu)  
);
```

```
CREATE TABLE Klient
(
id_klienta NUMERIC CONSTRAINT klient_pk PRIMARY KEY ,
imie VARCHAR(20),
nazwisko VARCHAR(25),
data_urodzenia DATE,
id_adresu NUMERIC CONSTRAINT klient_adres_fk REFERENCES Adres(id_adresu)
);
```

```
CREATE TABLE Transakcja
(
id_transakcji NUMERIC CONSTRAINT transakcja_pk PRIMARY KEY ,
id_klienta NUMERIC CONSTRAINT transakcja_klient_fk REFERENCES Klient(id_klienta),
id_pracownika NUMERIC CONSTRAINT transakcja_pracownik_fk REFERENCES Pracownik(id_pracownika),
id_filmu NUMERIC CONSTRAINT transakcja_film_fk REFERENCES Film(id_filmu),
data_transakcji DATE,
typ VARCHAR(10) CONSTRAINT typ_ck CHECK ( typ IN ('wypozyczenie','zwrot'))
);
```