

Dokumentacja projektu z przedmiotu Bazy Danych I

Baza produkcji filmowych pt. „Filmoholik”

Jakub Strugała
AGH
Wydział Fizyki i Informatyki Stosowanej
Informatyka Stosowana
19.01.2021

Plik wykonywalny projektu *Filmoholik.jar* znajduje się na serwerze Pascal w folderze pod linkiem:

http://pascal.fis.agh.edu.pl/~8strugala/BD1_Projekt/

1. Projekt koncepcji, założenia

1.1 Zdefiniowanie tematu projektu: krótki opis zadania, zdefiniowanie jego celów i zadań jakie ma realizować.

Tematem niniejszego projektu jest stworzenie aplikacji bazodanowej, która zawiera bazę filmową. Celem aplikacji jest ułatwienie użytkownikowi dostępu do szczegółowych informacji o poszczególnych produkcjach filmowych, czy też o osobach uczestniczących w tworzeniu tych produkcji.

Aplikacja pozwala na wyszukiwanie filmów i osób związanych z produkcjami filmowymi, sortowanie wyników wedle poszczególnych filtrów, dodawanie nowych produkcji czy osób oraz ocenianie filmów/seriali.

Inspiracją do projektu były takie serwisy jak IMDb.com czy też filmweb.pl.

1.2 Analiza wymagań użytkownika: określenie funkcjonalności jakie ma spełniać projektowana baza danych.

Aplikacja umożliwia:

- wyszukiwanie produkcji filmowej na podstawie nazwy
- wyszukiwanie osób związanych z filmem na podstawie imienia i nazwiska
- sortowanie wyników wedle poszczególnych filtrów
- rejestrację i logowanie

*tylko dla zalogowanych użytkowników:

- dodawanie nowych produkcji wraz z informacjami o nich
- dodawanie nowych osób wraz z informacjami o nich
- dodawanie oceny do danej produkcji filmowej

1.3 Zaprojektowanie funkcji: określenie podstawowych funkcji realizowanych w bazie danych.

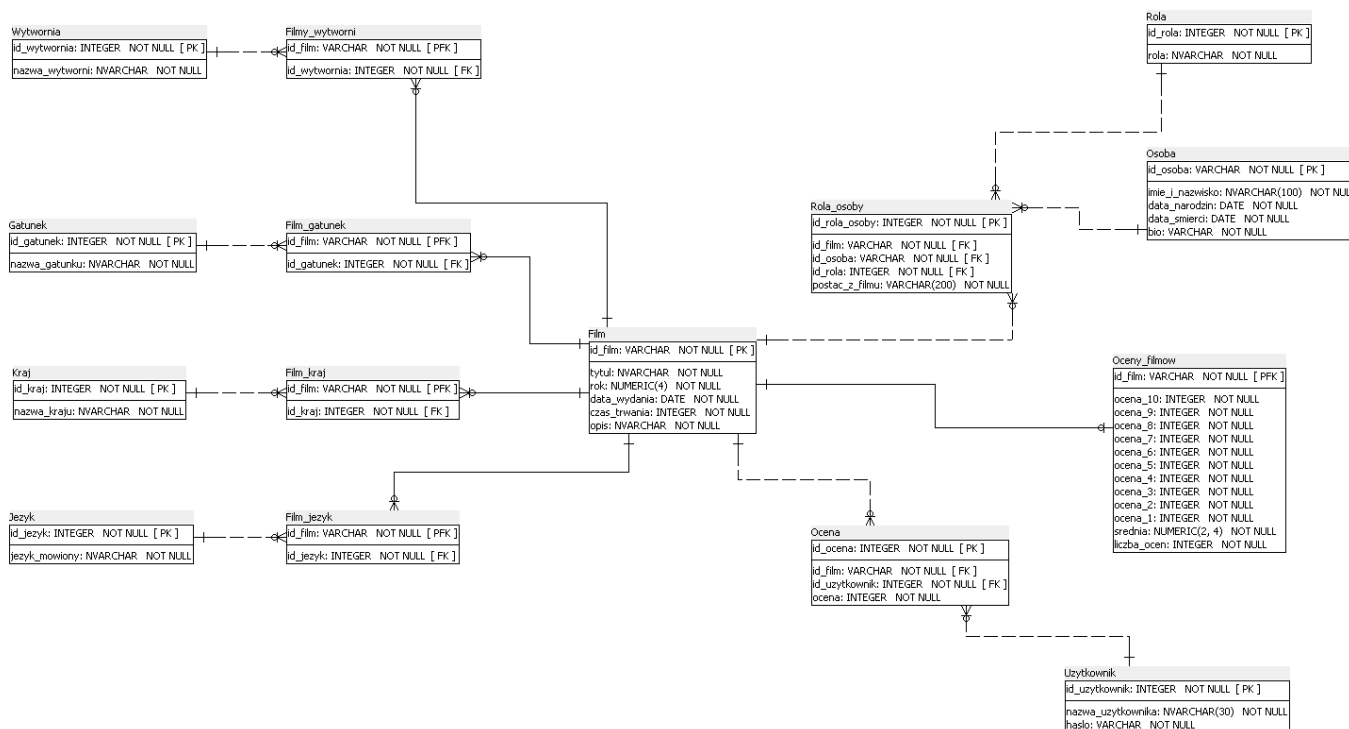
Podstawowe funkcje:

- dodające rekordy(np. nowe produkcje, nowi aktorzy) do wyjściowych baz danych
- dodające oceny do danej produkcji
- zwracające informacje o danej produkcji/osobie wraz z plakatem lub zdjęciem
- dodające ocenę do danej produkcji filmowej

- sortujące zwrócone wyniki wedle określonych kryteriów

2. Projekt diagramów (konceptualny)

Diagram ERD:



3. Projekt logiczny

3.1 Projektowanie tabel kluczy oraz indeksów

Widoczną na diagramie ERD bazę danych można podzielić na kilka istotnych części.

Centralną częścią bazy danych jest tabela **Film**, z którą poszczególne pozostałe tabelę znajdują się w odpowiednich relacjach, dlatego też na powyższym rysunku znajdują się ona na środku. Tabela ta zawiera najbardziej istotne informacje o filmie/serialu.

Po lewej stronie od tabeli **Film** znajduje się kolejna część, czyli 8 tabel reprezentujących pozostałe informacje o danej produkcji filmowej (4 z tych tabel są asocjacyjne).

Mianowicie, są to: **gatunek**, do którego należy dana produkcja, **język**, w którym mówią aktorzy tej produkcji, **miejsce kręcenia**, czyli tutaj - kraj oraz wytwórnia, pod której nadzorem wyprodukowano ten film/serial.

Do tej części możemy również zaliczyć jedną tabelę widoczną na prawo od tabeli Film, mianowicie tabelę **Oceny_filmow**. Jest ona również swoistym dodatkiem do informacji o produkcjach. Zawiera informacje o sumarycznej liczbie głosów przyznanych danemu utworowi (kolumna: *liczba_ocen*), o liczbie odpowiadającej danej ocenie reprezentowanej przez nazwę kolumny (*ocena_10*, *ocena_9*, itd.) oraz o średniej arytmetycznej z tych głosów (kolumna: *srednia*).

Kolejną część stanowią 3 tabele widoczne w prawym górnym rogu diagramu, reprezentujące informacje o osobie związanej z filmami (mogą to być aktorzy, reżyserzy, producenci itd.). Są to zatem tabele: **Rola_osoby**, **Rola** oraz **Osoba**.

Ostatnią część stanowią 2 tabele w prawym dolnym rogu – są to tabele reprezentujące danego użytkownika oraz oceny, które wystawił danym produkcjom.

Szczegóły techniczne:

Poniżej przedstawiony jest kod SQL odpowiadający za utworzenie tabel i relacji pomiędzy nimi. Kod znajduje się w folderze „Skrypty SQL” w pliku **tabele.sql**.

Podając kod zgodnie z opisem powyższych części bazy, mamy:

- Podstawowe informacje o filmie - centralna część - utworzenie tabeli **Film**:

```
create table if not exists "Film"
(
    id_film        varchar(10) not null,
    tytul          varchar      not null,
    rok            integer       not null,
    data_wydania   varchar      not null,
    czas_trwania   integer,
    opis           varchar,
    constraint film_pk
        primary key (id_film)
);

alter table "Film"
    owner to npbkqkcz;
```

Każdy film/serial posiada id (id_film), do którego przyporządkowane są konkretne informacje o nim.

Id_film jest kluczem głównym tej tabeli (Primary Key – PK).

- Dodatkowe informacje o filmie - lewa część – utworzenie tabel **Gatunek**, **Jezyk**, **Wytownia**, **Kraj**:

```
create table if not exists "Gatunek"
(
    nazwa_gatunku varchar,
    id_gatunku     integer not null,
    constraint gatunek_pk
    primary key (id_gatunku)
);

alter table "Gatunek"
    owner to npbkqkcz;

create table if not exists "Jezyk"
(
    jezyk_mowiony varchar,
    id_jezyk       integer not null,
    constraint jezyk_pk
    primary key (id_jezyk)
);

alter table "Jezyk"
    owner to npbkqkcz;

create table if not exists "Kraj"
(
    id_kraj       integer not null,
    nazwa_kraju   varchar,
    constraint kraj_pk
    primary key (id_kraj)
);

alter table "Kraj"
    owner to npbkqkcz;

create table if not exists "Wytownia"
(
    nazwa_wytworni varchar,
    id_wytownia     integer not null,
    constraint wytownia_pk
    primary key (id_wytownia)
);

alter table "Wytownia"
    owner to npbkqkcz;
```

Każdemu id tabeli odpowiada dana nazwa (gatunku, języka, kraju, wytwórni).

Są to zatem tzw. Tablice asocjacyjne czy też tabele słownikowe.

Tabele stoją z tabelą Film w relacji wiele do wielu (N:M).

- Dodatkowe informacje o filmie – lewa część - utworzenie tabel łącznikowych **Film_gatunek**, **Film_kraj**, **Film_jezyk**, **Film_wytwornia**:

```
create table if not exists "Film_gatunek"
(
    id_gatunku integer,
    id_film     varchar(10) not null,
    constraint film_gatunek_pk
        primary key (id_film),
    constraint id_gatunku
        foreign key (id_gatunku) references "Gatunek"
            on update cascade on delete cascade,
    constraint id_film
        foreign key (id_film) references "Film"
            on update cascade on delete cascade
);

alter table "Film_gatunek"
    owner to npbkqkcz;

create table if not exists "Film_jezyk"
(
    id_film     varchar(10) not null,
    id_jezyk    integer     not null,
    constraint film_jezyk_pk
        primary key (id_film),
    constraint id_film
        foreign key (id_film) references "Film"
            on update cascade on delete cascade,
    constraint id_jezyk
        foreign key (id_jezyk) references "Język"
            on update cascade on delete cascade
);

alter table "Film_jezyk"
    owner to npbkqkcz;
```

```

create table if not exists "Film_kraj"
(
    id_film varchar(10) not null,
    id_kraj integer,
    constraint film_kraj_pk
        primary key (id_film),
    constraint id_film
        foreign key (id_film) references "Film"
            on update cascade on delete cascade,
    constraint id_kraj
        foreign key (id_kraj) references "Kraj"
            on update cascade on delete cascade
);

alter table "Film_kraj"
    owner to npbkqkcz;

create table if not exists "Filmy_wytworni"
(
    id_film          varchar(10) not null,
    id_wytwornia integer,
    constraint filmy_wytworni_pk
        primary key (id_film),
    constraint id_film
        foreign key (id_film) references "Film"
            on update cascade on delete cascade,
    constraint id_wytwornia
        foreign key (id_wytwornia) references "Wytwornia"
            on update cascade on delete cascade
);

alter table "Filmy_wytworni"
    owner to npbkqkcz;

```

*Tabele te łączą się z tabelą **Film** za pomocą id_film, a następnie łączą się z poszczególnymi tabelami po lewej stronie za pomocą drugiego wspólnego id - kolejno: id_gatunku, id_jezyk, id_kraj, id_wytwornia.*

Wszystkie kolumny tych tabel stanowią zatem klucze obce – Foreign Key (FK).
Tabele te służą do zamodelowania związku „wiele do wielu” tabel **Film** – **Gatunek**, **Kraj**,
Jezyk, **Wytownia**, przy pomocy dwóch relacji 1:N.

- Dodatkowe informacje o filmie – tabela **Oceny_filmow**:

```
create table if not exists "Oceny_filmow"
(
    id_film        varchar(10) not null,
    ocena_10       integer,
    ocena_9        integer,
    ocena_8        integer,
    ocena_7        integer,
    ocena_6        integer,
    ocena_5        integer,
    ocena_4        integer,
    ocena_3        integer,
    ocena_2        integer,
    ocena_1        integer,
    srednia        numeric(4, 2),
    liczba_ocen    integer,
    constraint oceny_filmow_pk
        primary key (id_film),
    constraint id_film
        foreign key (id_film) references "Film"
            on update cascade on delete cascade
);

alter table "Oceny_filmow"
    owner to npbkqkcz;
```

Tabela ta łączy się z tabelą **Film** przy użyciu wspólnego id - `id_film`, gdzie dla tabeli **Oceny_filmow** `id_film` stanowi Primary Foreign Key (PFK).
Stoi ona w relacji 1:1 z tabelą **Film**, gdyż każdemu `id_film` z tabeli **Film** odpowiada ten sam `id_film` z tabeli **Oceny_filmow**.

- Kolejna część, czyli informacje o osobach – prawy górny róg – tabele **Rola**, **Osoba**, **Rola_osoby**:

```
create table if not exists "Rola"
(
    rola          varchar(20),
    id_rola       integer not null,
    constraint rola_pk
        primary key (id_rola)
);

alter table "Rola"
    owner to npbkqkcz;
```



```

create table if not exists "Osoba"
(
    id_osoba          varchar(15) not null,
    imie_i_nazwisko  varchar,
    data_narodzin     date,
    data_smierci      date,
    bio               varchar,
    constraint osoba_1_pk
        primary key (id_osoba)
);

```

Każdemu id_rola odpowiada nazwa danego zawodu,

```

alter table "Osoba"
    owner to npbkqkcz;

```

przykładowo: „actress”, „writer” itp. Jest to zatem encja słownikowa – zawiera unikatowy klucz i wartość mu przypisaną.

Każdemu id_osoba odpowiadają szczegółowe informacje o danej osobie, widniejącej pod tym id.

```

Tabela create table if not exists "Rola_osoby"
(
    id_film          varchar(10) ,
    id_osoba         varchar(15) ,
    id_rola          integer,
    postac_z_filmu   varchar,
    id_rola_osoby    serial not null,
    constraint rola_osoby_pk
        primary key (id_rola_osoby) ,
    constraint id_film
        foreign key (id_film) references "Film"
            on update cascade on delete cascade,
    constraint id_osoba
        foreign key (id_osoba) references "Osoba"
            on update cascade on delete cascade,
    constraint id_rola
        foreign key (id_rola) references "Rola"
            on update cascade on delete cascade
);

alter table "Rola_osoby"
    owner to npbkqkcz;

```

Rola_osoby jest tabelą łączącą tabele **Film**, **Osoba** i **Rola**, gdzie **id_rola_osoby** stanowi klucz główny tej tabeli.

Pole **postac_z_filmu** jest puste dla każdej roli różnej od „actress” albo „actor”.

Z racji, że tabela **Film** i **Osoba** są w relacji N:M (dana osoba może wystąpić w wielu filmach oraz w danym filmie może wystąpić wiele różnych osób) tabela **Rola_osoby** łączy je przy pomocy dwóch relacji 1:N.

- Ostatnia część, czyli informacje o użytkowniku i jego ocenach – prawy dolny róg – tabele **Uzytkownik** i **Ocena**:

```

create table if not exists "Uzytkownik"
(
    id_uzytkownik    integer not null,
    nazwa_uzytkownika varchar(30) ,
    haslo            varchar not null,
    constraint uzytkownik_pk
        primary key (id_uzytkownik)
);

alter table "Uzytkownik"
    owner to npbkqkcz;

```

Danemu **id_uzytkownik** przypisana jest nazwa użytkownika i hasło, gdzie **id_uzytkownik** jest kluczem głównym tej tabeli.

```

create table if not exists "Ocena"
(
    id_film          varchar(10) not null,
    id_uzytkownik    integer      not null,
    ocena            integer,
    id_ocena          serial       not null,
    constraint ocena_pk
        primary key (id_ocena),
    constraint id_film
        foreign key (id_film) references "Film"
            on update cascade on delete cascade,
    constraint id_uzytkownik
        foreign key (id_uzytkownik) references "Uzytkownik"
            on update cascade on delete cascade
);

alter table "Ocena"
    owner to npbkqkcz;

create unique index if not exists ocena_id_ocena_uindex
    on "Ocena" (id_ocena);

```

Tabela **Ocena** łączy tabele **Film** i **Uzytkownik** za pomocą `id_film` oraz `id_uzytkownik`. Kluczem głównym tej tabeli jest `id_ocena`, natomiast jedyne możliwe wartości, które przyjmuje kolumna `ocena` to wartości od 1 – 10, co wymuszone jest dopiero na poziomie aplikacji. Relacja **Ocena** realizuje łączenie tabel **Film** oraz **Uzytkownik** (których relacja jest wiele do wielu -N:M) za pomocą dwóch relacji 1:N. Dany użytkownik może dać ocenę wielu filmom oraz dany film może być oceniony przez wielu użytkowników.

3.2 Słownik danych – opis nieoczywistych danych i ich ograniczeń:

Film:

- **id_film** – unikatowe ID filmu, jest typu VARCHAR z racji importowania bazy udostępnianej przez Internet Movie Database (IMDb.com), z której zarówno ID produkcji, jak i ID aktorów są w formie tekstowej – przykładowo: „tt8901582”. Posiada maksymalną długość ustawioną na 10 znaków gdyż obecnie długość ta wynosi 9, zatem 10-ty znak stanowi zapas.

Rola_osoby:

- **postac_z_filmu** – pole to odpowiada za nazwę postaci, granej przez danego aktora/aktorkę. Dla osób posiadających przypisany im zawód inny niż „actor” czy „actress” (a więc `id_rola` różne od 1 lub 2), pole to jest równe *null*.

Oceny_filmow:

- **ocena_10, ocena_9, itd.** - kolumny o takich nazwach zawierają pod sobą sumaryczną liczbę głosów użytkowników o wartości równej tej znajdującej się w nazwie kolumny. Zatem przykładowo kolumna *ocena_10* dla danego *id_film* ma pod sobą liczbę ocen oddanych na ten film przez użytkowników, których wartość (czy też waga oceny) wynosi 10. Analogicznie z pozostałymi kolumnami.

- **liczba_ocen** – kolumna ta odpowiada sumarycznej liczbie ocen ze wszystkich kolumn od *ocena_1* do *ocena_10* oddanych na poszczególny film.

Ocena:

- **ocena** – ocena od 1 do 10 wystawiana danemu filmowi przez danego użytkownika, zakres 1-10 jest wymuszany na poziomie aplikacji.

3.3 Analiza zależności funkcyjnych i normalizacja tabel

Większość tabel spełnia warunki zakładanych postaci normalnych, jednakże występują pewne „anomalie” czy też niedoskonałości, których uzasadnienie znajduje się w punkcie 3.4.

1NF

Tabele spełniają 1NF, gdyż każda z nich opisuje jeden obiekt, kolejność wierszy może być dowolna, bo znaczenie danych nie zależy od kolejności, (większość) relacji posiada wartości atrybutów, które są niepodzielne, czyli atomowe oraz (większość) relacji nie zawiera kolekcji, a więc powtarzających się grup informacji. Wyjątki od wspomnianych założeń opisane są w punkcie 3.4.

2NF

Zakładając, że tabele spełniają pierwszą postać normalną, będą spełniać drugą, gdy żadna niekluczowa kolumna nie jest częściowo funkcyjnie zależna od klucza potencjalnego (a więc w przypadku tego projektu od ID danej tabeli). Z racji, że każdy niekluczowy atrybut (czyli kolumna) jest w całości zależna od całego klucza potencjalnego a więc od ID danej tabeli, relacje spełniają drugą postać normalną.

3NF

Biorąc pod uwagę fakt, że każdy atrybut niekluczowy jest zależny jedynie od atrybutów kluczowych – ID danej kolumny, relacje spełniają trzecią postać normalną.

3.4 Denormalizacja struktury tabel

Z racji, że dane użyte do realizacji projektu pochodziły z zewnętrznych źródeł mianowicie z bazy IMDb (Internet Movie Database), sposób w jaki zostały wypełnione w plikach .csv definiował często odgórnie relacje pomiędzy tabelami. Przykładowo przy projektowaniu tabel: **Gatunek**, **Jezyk** czy **Kraj** do danego id_filmu przypisane zostały ciągi znaków:

„Drama, Thriller, Comedy”, podobnie z tabelą **Jezyk**: „English, French, German” i analogicznie z tabelą **Kraj**.

Wiele trudu zajęłoby zatamizowanie tych ciągów do osobnych wartości (przykładowo: „English”, „French”, „German”) i jednocześnie ich połączenie z danymi ID produkcji, toteż do zminimalizowania problemu pobrane dane uporządkowane zostały przed zaimportowania wedle unikatowych ciągów takich jak: „English, French, German”, „English, German”, „English, French” itd. I każdemu z nich przypisane osobne ID, gdyż faktycznie każdy ciąg takich znaków różni się od innego, przez co możemy go uznać za unikatowy.

Z uwagi na to, że taka jest specyfika danych dostarczanych przez IMDb, tak przygotowane dane zostały zaimportowane do projektu, stąd mogące wystąpić niektóre anomalie i niewielkie niespójności danych.

Przykładowe polecenia służące usuwaniu konkretnej liczby rekordów oraz danych niepowiązanych odpowiednio pomiędzy tabelami:

1) usuwanie konkretnej liczby rekordów przy użyciu mechanizmu ctid:

```
--Usuwanie danej liczby rekordow z bazy za pomoca ctid
-- dzieki ustawieniu ON DELETE CASCADE na pozostalych bazach, usuniecie powtarza sie automatycznie w nich
DELETE
FROM "Film"
WHERE ctid IN (
    SELECT ctid
    FROM "Film"
    ORDER BY id_film
    LIMIT 4700
);
```

(na powyższym obrazku przykładowa liczba rekordów do usunięcia wynosi 4700)

2) usuwanie niepowiązanych odpowiednio rekordów:

```
-- usuwanie rekordow z tabeli Oceny_filmow niepowiazanych z tabela Film
DELETE
FROM "Oceny_filmow" fg
WHERE NOT EXISTS(SELECT *
                  FROM "Film" f
                  WHERE f.id_film = fg.id_film
                  );
```

3.5 Zaprojektowanie operacji na danych

Wybrane operacje wykonywane na danych:

- **Triggery i funkcje zwracające triggery** (w folderze „Skrypty SQL” pliku *funkcje_trigger.sql*):

1) funkcja dodaj()

```

create or replace function dodaj() returns trigger
    language plpgsql
as
$$
BEGIN
    RAISE NOTICE 'Nowa ocena: %', NEW.ocena;
    IF NEW.ocena = 1 THEN
        UPDATE "Oceny_filmow" set ocena_1 = ocena_1 + 1 where id_film = NEW.id_film;
    ELSIF NEW.ocena = 2 THEN
        UPDATE "Oceny_filmow" set ocena_2 = ocena_2 + 1 where id_film = NEW.id_film;
    ELSIF NEW.ocena = 3 THEN
        UPDATE "Oceny_filmow" set ocena_3 = ocena_3 + 1 where id_film = NEW.id_film;
    ELSIF NEW.ocena = 4 THEN
        UPDATE "Oceny_filmow" set ocena_4 = ocena_4 + 1 where id_film = NEW.id_film;
    ELSIF NEW.ocena = 5 THEN
        UPDATE "Oceny_filmow" set ocena_5 = ocena_5 + 1 where id_film = NEW.id_film;
    ELSIF NEW.ocena = 6 THEN
        UPDATE "Oceny_filmow" set ocena_6 = ocena_6 + 1 where id_film = NEW.id_film;
    ELSIF NEW.ocena = 7 THEN
        UPDATE "Oceny_filmow" set ocena_7 = ocena_7 + 1 where id_film = NEW.id_film;
    ELSIF NEW.ocena = 8 THEN
        UPDATE "Oceny_filmow" set ocena_8 = ocena_8 + 1 where id_film = NEW.id_film;
    ELSIF NEW.ocena = 9 THEN
        UPDATE "Oceny_filmow" set ocena_9 = ocena_9 + 1 where id_film = NEW.id_film;
    ELSIF NEW.ocena = 10 THEN
        UPDATE "Oceny_filmow" set ocena_10 = ocena_10 + 1 where id_film = NEW.id_film;
    END IF;

    RETURN NEW;
END;
$$;

alter function dodaj() owner to npbkqkcz;

```

Funkcja *dodaj()* realizuje UPDATE na tabeli **Oceny_filmow**, dodając do danej wartości kolumny (od *ocena_1* do *ocena_10*) wartość o jeden większą z racji ocenienia danej produkcji przez użytkownika.

Funkcja jest uruchamiana przed wstawieniem i przed aktualizowaniem tabeli Ocena: (z pliku **tabele.sql**, linia: **173**)

```

create trigger dodajtrigger
    before insert
    on "Ocena"
    for each row
execute procedure dodaj();

create trigger dodajtrigger2
    before update
    on "Ocena"
    for each row
execute procedure dodaj();

```

Przed wstawieniem uruchamiana jest, aby zaktualizować wartość danej kolumny, co potrzebne jest później do obliczenia średniej ocen konkretnego filmu/serialu.

Natomiast przed zaktualizowaniem uruchamiana na wypadek, gdyby użytkownik chciał zmienić ocenę, którą już wcześniej wybrał dla danej produkcji (a więc nadpisać istniejącą, a nie dodać nową).

2) funkcja odejmij()

```
create or replace function odejmij() returns trigger
    language plpgsql
as
$$
BEGIN
    RAISE NOTICE 'Stara ocena: %', OLD.ocena;
    RAISE NOTICE 'Nowa ocena: %', NEW.ocena;
    IF OLD.ocena = 1 THEN
        UPDATE "Oceny_filmow" set ocena_1 = ocena_1 - 1 where id_film = NEW.id_film;
    ELSIF OLD.ocena = 2 THEN
        UPDATE "Oceny_filmow" set ocena_2 = ocena_2 - 1 where id_film = NEW.id_film;
    ELSIF OLD.ocena = 3 THEN
        UPDATE "Oceny_filmow" set ocena_3 = ocena_3 - 1 where id_film = NEW.id_film;
    ELSIF OLD.ocena = 4 THEN
        UPDATE "Oceny_filmow" set ocena_4 = ocena_4 - 1 where id_film = NEW.id_film;
    ELSIF OLD.ocena = 5 THEN
        UPDATE "Oceny_filmow" set ocena_5 = ocena_5 - 1 where id_film = NEW.id_film;
    ELSIF OLD.ocena = 6 THEN
        UPDATE "Oceny_filmow" set ocena_6 = ocena_6 - 1 where id_film = NEW.id_film;
    ELSIF OLD.ocena = 7 THEN
        UPDATE "Oceny_filmow" set ocena_7 = ocena_7 - 1 where id_film = NEW.id_film;
    ELSIF OLD.ocena = 8 THEN
        UPDATE "Oceny_filmow" set ocena_8 = ocena_8 - 1 where id_film = NEW.id_film;
    ELSIF OLD.ocena = 9 THEN
        UPDATE "Oceny_filmow" set ocena_9 = ocena_9 - 1 where id_film = NEW.id_film;
    ELSIF OLD.ocena = 10 THEN
        UPDATE "Oceny_filmow" set ocena_10 = ocena_10 - 1 where id_film = NEW.id_film;
    END IF;
    RETURN NEW;
END;
$$;

alter function odejmij() owner to npbkqkcz;
```

Funkcja *odejmij()* realizuje UPDATE na tabeli **Oceny_filmow**, odejmując od danej wartości kolumny (od *ocena_1* do *ocena_10*) wartość równą 1 w przypadku, gdy użytkownik chce zmienić ocenę dla produkcji, dla której już kiedyś ocenę wybrał.

Kluczową rzeczą jest tutaj fakt, iż w przeciwieństwie do funkcji *dodaj()*, w tym wypadku korzystamy nie ze specjalnej zmiennej **NEW**, lecz ze zmiennej **OLD**, dzięki której dowiemy się, która z ocen została wstawiona przed nowym wyborem i dekrementujemy właśnie tę liczbę ocen dla tej starej oceny.

Dzięki temu sumaryczna liczba wszystkich ocen dla danej produkcji nie zmienia się, jeżeli użytkownik jedynie zmienia ocenę, jaką niegdyś dał konkretnemu filmowi.

Funkcja *odejmij()* jest uruchamiana przed aktualizowaniem tabeli Ocena (z pliku *tabele.sql*, linia: 185):

```
create trigger odejmijtrigger
    before update
    on "Ocena"
    for each row
    execute procedure odejmij();
```

3) funkcja policzsredniaocen()

```
create or replace function policzsredniaocen() returns trigger
language plpgsql
as
$$
BEGIN
    NEW.liczba_ocen = NEW.ocena_10 + NEW.ocena_9 + NEW.ocena_8 + NEW.ocena_7 + NEW.ocena_6 +
        NEW.ocena_5 + NEW.ocena_4 + NEW.ocena_3 + NEW.ocena_2 +
        NEW.ocena_1;
    IF NEW.liczba_ocen = 0 THEN
        NEW.liczba_ocen = 1;
    END IF;

    NEW.srednia :=
        (SELECT (NEW.ocena_10 * 10 + NEW.ocena_9 * 9 + NEW.ocena_8 * 8 + NEW.ocena_7 * 7 +
            NEW.ocena_6 * 6 +
            NEW.ocena_5 * 5 + NEW.ocena_4 * 4 + NEW.ocena_3 * 3 + NEW.ocena_2 * 2 +
            NEW.ocena_1 * 1) /
            CAST(NEW.liczba_ocen AS NUMERIC(4, 1))
        FROM "Oceny_filmow"
        where id_film = NEW.id_film);

    RAISE NOTICE 'Srednia nowa: %, Srednia stara: %, id_film: %', NEW.srednia, OLD.srednia, NEW.id_film;
    RAISE NOTICE 'Liczba ocen nowa: %, Liczba ocen stara: %, id_film: %', NEW.liczba_ocen, OLD.liczba_ocen, NEW.id_film;
    RETURN NEW;
END ;
$$;

alter function policzsredniaocen() owner to npbkqkcz;
```

Funkcja *policzsredniaocen()* aktualizuje zarówno liczbę ocen dla tabeli **Oceny_filmow**, jak i średnią arytmetyczną.

Uruchamiana jest przed aktualizacją danych na tabeli **Oceny_filmow**.

Natomiast faktyczne dodawanie oceny do tabeli

```
create trigger policzsredniaocenttrigger
before update
on "Oceny_filmow"
for each row
execute procedure policzsredniaocen();
```

Ocena oraz UPDATE realizowane jest w funkcji *handle()* w pliku **Controller.java** w liniach kolejno **930** oraz **946**.

- **Widoki** (w folderze „Skrypty SQL” w pliku **widoki.sql**):

1) osoba_info


```

create or replace view osoba_info
    (id_osoba, imie_i_nazwisko, data_narodzin, data_smierci, bio, rola, tytul, rok,
     postac_z_filmu) as
SELECT o.id_osoba,
       o.imie_i_nazwisko,
       o.data_narodzin,
       o.data_smierci,
       o.bio,
       r.rola,
       f.tytul,
       f.rok,
       ro.postac_z_filmu
FROM "Osoba" o
      JOIN "Rola_osoby" ro ON o.id_osoba::text = ro.id_osoba::text
      JOIN "Rola" r ON r.id_rola = ro.id_rola
      JOIN "Film" f ON ro.id_film::text = f.id_film::text
ORDER BY o.imie_i_nazwisko;

alter table osoba_info
    owner to npbkqkcz;

```

Wykorzystywany w funkcji `zwrocOsobeNowe()` z pliku ***Controller.java***.

Służy do zwrócenia szczególnych informacji z różnych tabel dotyczących osoby (i produkcji, w której grała), które łączą się ze sobą kolumnami wymienionymi w operacji **JOIN (INNER JOIN)**.

W tym przypadku widok ***osoba_info*** służy do zwrócenia szczegółowych informacji o danej osobie, które wyświetlane są w aplikacji pod zakładką „Szukaj osób filmu” po wyszukaniu danej osoby.

2) film_info

```

create or replace view film_info
    (id_film, tytul, rok, srednia, liczba_ocen, opis, nazwa_gatunku, nazwa_kraju,
    data_wydania, czas_trwania, nazwa_wytworni, jezyk_mowiony, imie_i_nazwisko, id_rola)
as
SELECT f.id_film,
       f.tytul,
       f.rok,
       oc.srednia,
       oc.liczba_ocen,
       f.opis,
       g.nazwa_gatunku,
       k.nazwa_kraju,
       f.data_wydania,
       f.czas_trwania,
       w.nazwa_wytworni,
       j.jezyk_mowiony,
       o.imie_i_nazwisko,
       ro.id_rola
FROM "Film" f
    JOIN "Film_kraj" fk USING (id_film)
    JOIN "Kraj" k USING (id_kraj)
    JOIN "Film_gatunek" fg ON f.id_film::text = fg.id_film::text
    JOIN "Gatunek" g USING (id_gatunku)
    JOIN "Film_jezyk" fj ON f.id_film::text = fj.id_film::text
    JOIN "Jezyk" j USING (id_jezyk)
    JOIN "Filmy_wytworni" fw ON f.id_film::text = fw.id_film::text
    JOIN "Wytownia" w USING (id_wytownia)
    LEFT JOIN "Rola_osoby" ro ON
CASE
    WHEN ro.id_film::text = f.id_film::text AND
         (ro.id_rola = 1 OR ro.id_rola = 2 OR ro.id_rola = 3) AND ro.id_rola IS NOT NULL
    THEN true
    ELSE false
END
    LEFT JOIN "Osoba" o ON
CASE
    WHEN ro.id_osoba::text = o.id_osoba::text AND o.id_osoba IS NULL THEN false
    WHEN ro.id_osoba::text = o.id_osoba::text AND o.id_osoba IS NOT NULL THEN true
    ELSE NULL::boolean
END
    LEFT JOIN "Oceny_filmow" oc ON f.id_film::text = oc.id_film::text;

alter table film_info
    owner to npbkqkcz;

```

Wykorzystywany w funkcji `zwrocFilmNowe()` z pliku **Controller.java**.

Służy do zwrócenia szczególnych informacji z różnych tabel dotyczących filmu/serialu, które łączą się ze sobą kolumnami wymienionymi w operacji **JOIN (INNER JOIN)** oraz **LEFT JOIN**.

Konstrukcja **CASE** została użyta, aby zwrócić jedynie osoby będące aktorami i reżysera bez pozostałych funkcji.

W tym przypadku widok **film_info** służy do zwrócenia szczegółowych informacji o danym filmie, które wyświetlane są w aplikacji pod zakładką „Szukaj filmu” po wyszukaniu danej produkcji.

W kodzie aplikacji znajdują się również liczne podstawowe kwerendy typu **SELECT**, **INSERT** czy **UPDATE**, odpowiadające za podstawowe operacje na bazie danych z poziomu aplikacji.

4. Projekt funkcjonalny

4.1 Interfejsy do prezentacji, edycji i obsługi danych

Zwrócone dane wyświetlane są podczas działania aplikacji na głównym ekranie pod polem tekstowym. Do wyświetlenia plakatów i zdjęć osób projekt wykorzystuje API The Movie Database (TMDb), do którego zapytania są wysyłane na podstawie unikatowych ID aktorów i ID filmów, które baza TMDb czerpie również z IMDb. Z tego powodu, możliwe jest wyświetlenie faktycznych zdjęć aktorów, czy filmów gdyż bazy IMDb i TMDb posiadają wspólne atrybuty.

W ramach uznania praw autorskich API TMDb zamieszczone zostało logo API na środku w dolnym pasku aplikacji. Z wymienionego API czerpane są jedynie informacje o plakatach i zdjęciach osób z wykluczeniem wszelkich innych informacji.

Stworzono formularze do wprowadzania danych do wszystkich tabel, które dostępne są z poziomu aplikacji pod kolejnymi przyciskami: „Dodaj ocenę”, „Dodaj film”, „Dodaj osobę” oraz „Zatwierdź” po wejściu w formularz z rejestracją.

4.2 Wizualizacja danych

W ramach zapytań do bazy danych zwracane są informacje o filmach i osobach.

W ramach raportu o filmach zwracane są informacje o tytule, opisie, reżyserze, gatunku, kraju, języku, czasie trwania, roku produkcji, dacie wydania, liczbie głosów oddanych na film, średniej z głosów i obsadzie wraz z widniejącym plakatem, jeżeli takowy jest dostępny z poziomu API. Jeżeli obrazek nie zostaje znaleziony, załadowany zostaje obrazek zastępczy informujący o braku dostępnego plakatu dla filmu.

W ramach raportu o osobach zwrócone zostają informacje o imieniu i nazwisku, dacie narodzin, dacie śmierci, biografii, zawodzie i o produkcjach, przy których ta osoba pracowała. Załadowany zostaje również plakat, jeżeli ID osoby widnieje w API TMDb. Jeżeli nie, analogicznie do filmów – załadowany zostaje obrazek zastępczy informujący o braku dostępnego zdjęcia.

W obu przypadkach dane mogą być sortowane przez przyciski dostępne na głównym ekranie, widoczne po słowach: „Sortuj po”.

Przez fakt, iż zaimplementowana baza posiada blisko 1000 rekordów filmów oraz około 8000 rekordów aktorów wprowadzono ograniczenie w postaci wyświetlanych wyników dla celów wydajnościowych aplikacji. Ustawiona maksymalna liczba wyświetlanych wyników wynosi 5 zarówno dla osób jak i filmów. Z tego też powodu przyciski odpowiadające za sortowanie nie sortują jedynie wyświetlonych wyników lecz wszystkie wyniki, które zwrócone zostały z bazy, a więc wynik sortowania może się okazać nieintuicyjny niemniej jednak poprawny.

4.3 Zdefiniowanie panelu sterowania aplikacji

Panel sterowania umożliwia użytkownikowi szereg akcji. Poza oczywistymi jakimi jest przechodzenie pomiędzy zakładkami z wyszukiwarką filmów i osób czy też samo wyszukiwanie konkretnych produkcji filmowych i osób związanych z filmem możliwe jest

również m.in. sortowanie zwróconych wyników dostępne po wybraniu konkretnego sortowania i kliknięciu przycisku z lupą powiększającą.

Po zarejestrowaniu i zalogowaniu, które to operacje dostępne są po wejściu w odpowiednie przyciski na tzw. głównej scenie – kolejno: „*Rejestracja*” i „*Logowanie*”, użytkownik dostaje większy dostęp do bazy i od teraz może dodawać oceny do produkcji czy też nowe produkcje i nowe osoby.

Możliwe jest również wylogowanie z aplikacji, jej minimalizacja i zamknięcie.

Całość zobrazowana w ostatnim punkcie dokumentacji – w punkcie **5**.

4.4 Makropolecenia

Wykonywane na kliknięcie przycisków:

- „*Zatwierdź*” – znajduje się w formularzach z dodawaniem nowej osoby i nowego filmu; jego kliknięcie potwierdza decyzję użytkownika o dodaniu danej osoby/produkcji
- „*Zarejestruj*” – rejestruje nowego użytkownika a więc dodaje nowego użytkownika do bazy danych,
- „*Dodaj ocenę*” - dodaje nową ocenę do danej produkcji, która przypisana jest konkretnemu użytkownikowi.

5. Dokumentacja

5.1 Wprowadzanie danych

Pierwotne dane zostały wprowadzone poprzez importowanie przygotowanych plików .csv z danymi dla konkretnych tabel (pliki te znajdują się w folderze „*Import*”). Dane te zostały udostępnione przez Internet Movie Database, dla celów projektowych jedynie skrócone i nieco uporządkowane wedle potrzeb.

Natomiast dane wprowadzane przez użytkownika pobierane są z pól tekstowych, które wypełnia w danym formularzu, a następnie przekazywane do funkcji, które je przetwarzają i dodają w ramach poleceń **UPDATE / INSERT** do odpowiednich tabel.

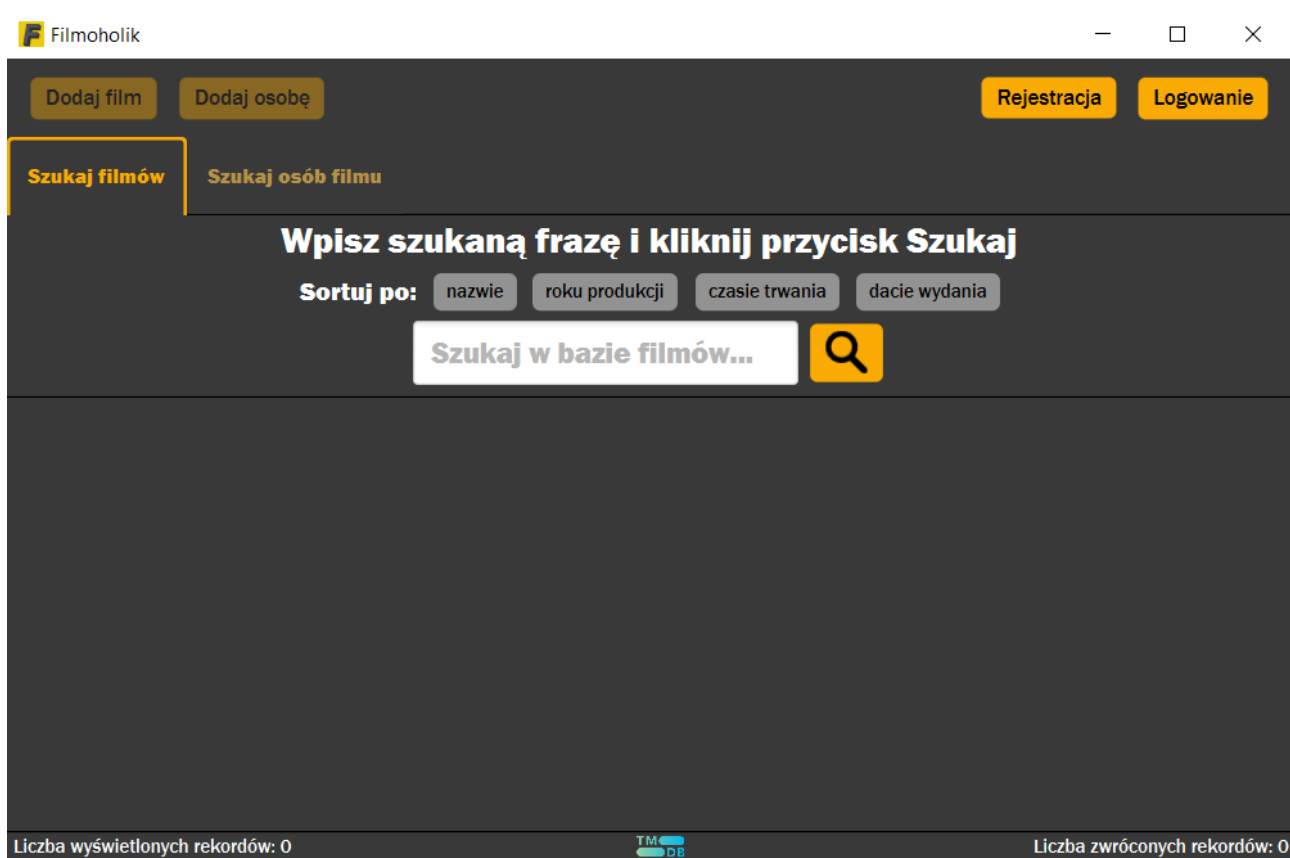
5.2 Dokumentacja użytkownika

Dla celów testowych utworzono próbnego użytkownika oraz próbną produkcję:

- Użytkownik o nazwie: „admin” i hasło: „admin”

- Produkcja o nazwie: „Nowy Film”

Po uruchomieniu aplikacji pojawia się następujący ekran:



W prawym górnym rogu widoczne są przyciski odpowiadające oczywiście za rejestrację i logowanie użytkownika. Oto okna pojawiające się na kliknięcie poszczególnych z nich:

- Rejestracja:

Filmoholik

Rejestracja

Wpisz nazwę użytkownika:

Wpisz hasło:

Zarejestruj ✓

Anuluj ✕

- Logowanie:

Filmoholik

Logowanie

Nazwa użytkownika:

Hasło:

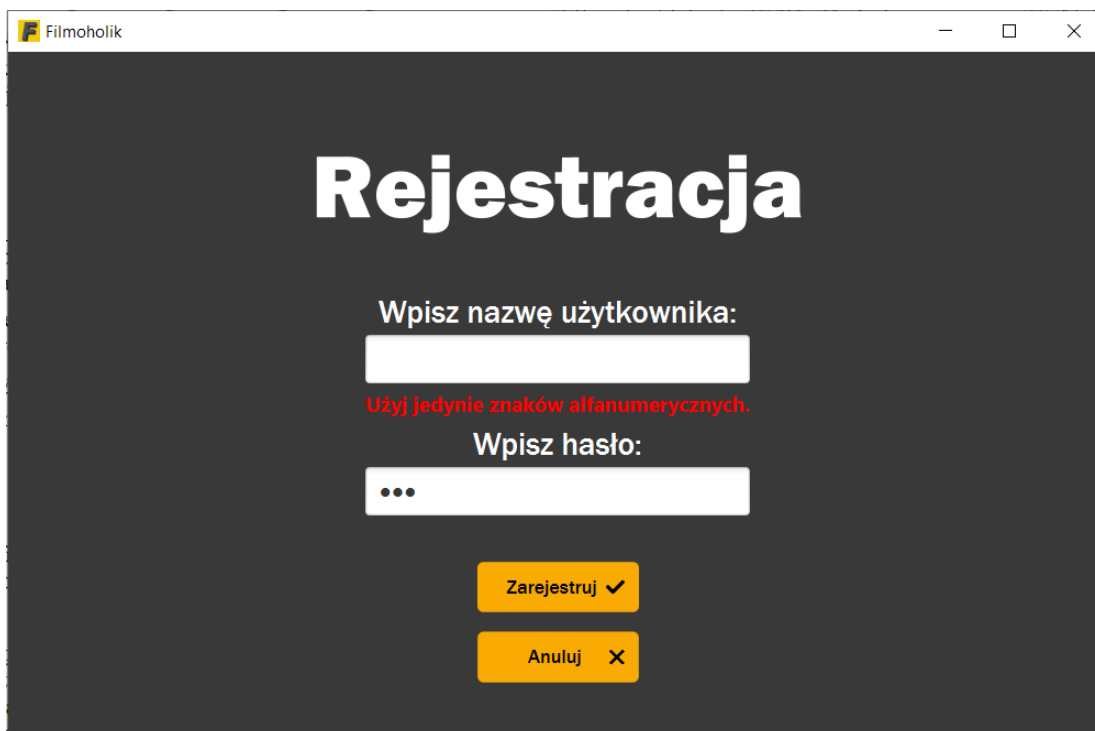
Zaloguj ✓

Anuluj ✕

Aby się zarejestrować, należy podać unikatową nazwę użytkownika. Jeżeli nazwa będzie niepoprawna bądź będzie istniała już taka nazwa w bazie, nowy użytkownik nie zostaje dodany a jedynie poproszony o wprowadzenie nowej nazwy.

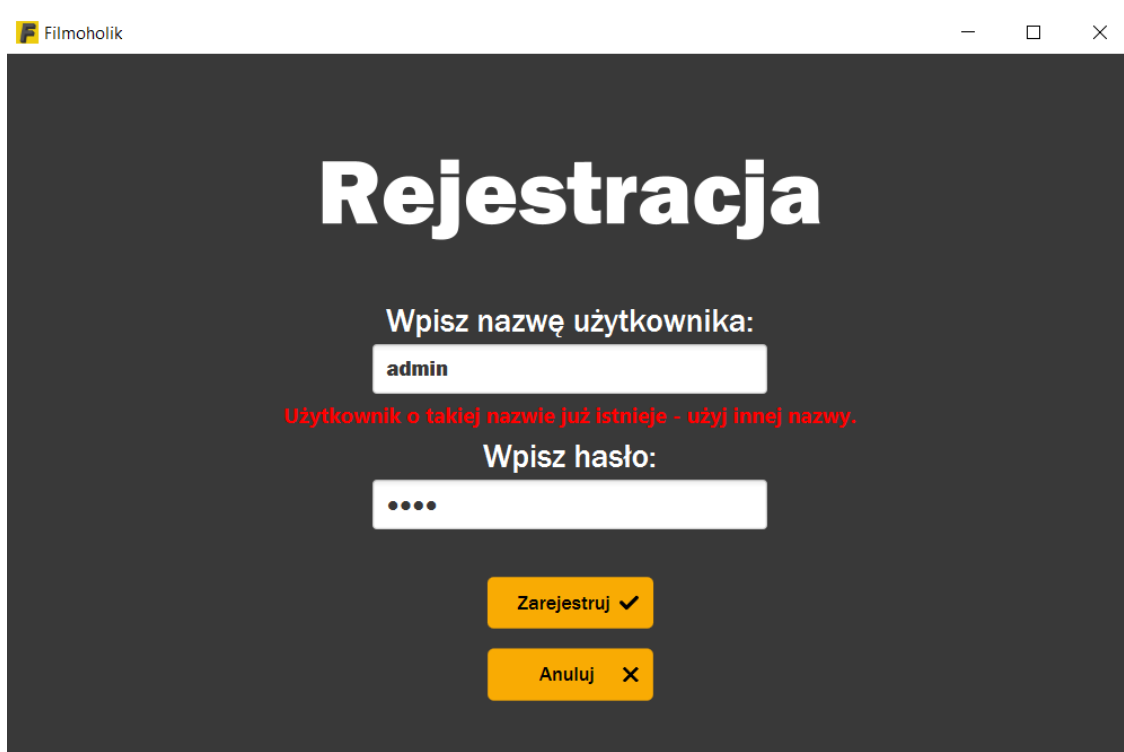
Przykładowe komunikaty, gdy nazwa jest niepoprawna bądź zajęta:

1) w nazwie wpisano same spacje:



The screenshot shows a web browser window titled "Filmoholik". The main heading is "Rejestracja" in large white letters on a dark background. Below it, the text "Wpisz nazwę użytkownika:" is followed by an empty white input field. A red error message "Użyj jedynie znaków alfanumerycznych." is displayed below the field. The text "Wpisz hasło:" is followed by a white input field with three dots indicating a password. At the bottom, there are two yellow buttons: "Zarejestruj ✓" and "Anuluj ✕".

2) nazwa jest już zajęta:

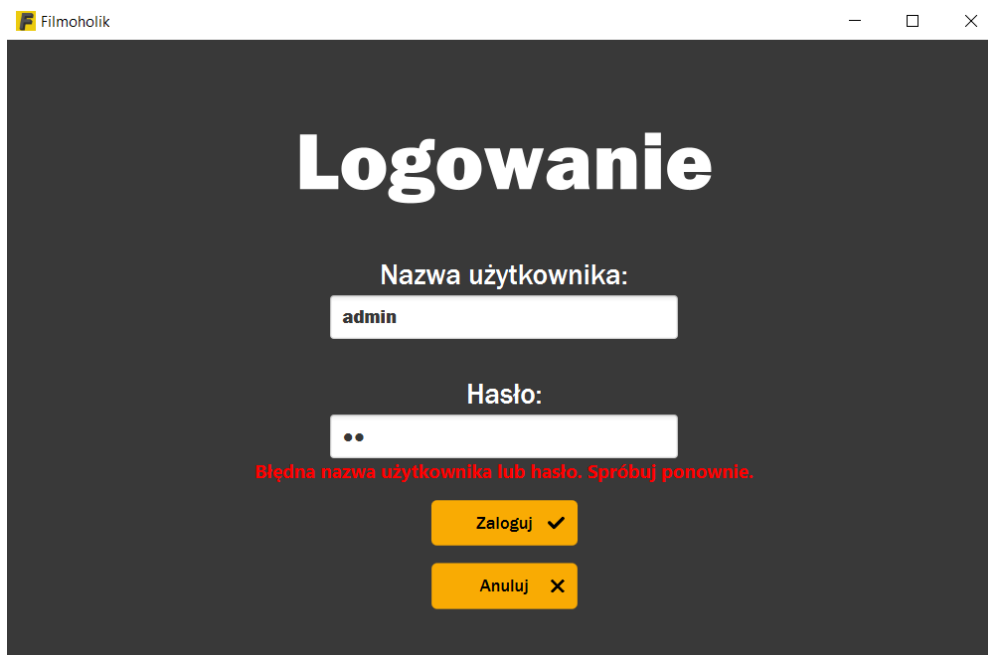


The screenshot shows the same "Rejestracja" page, but the username field now contains the text "admin". A red error message "Użytkownik o takiej nazwie już istnieje - użyj innej nazwy." is displayed below the field. The password field and buttons remain the same.

W przypadku gdy nazwa jest unikatowa i poprawna, na kliknięcie przycisku „Zarejestruj” zostanie dodany do bazy nowy użytkownik i logowanie z tą nazwą użytkownika i hasłem zostanie udostępnione.

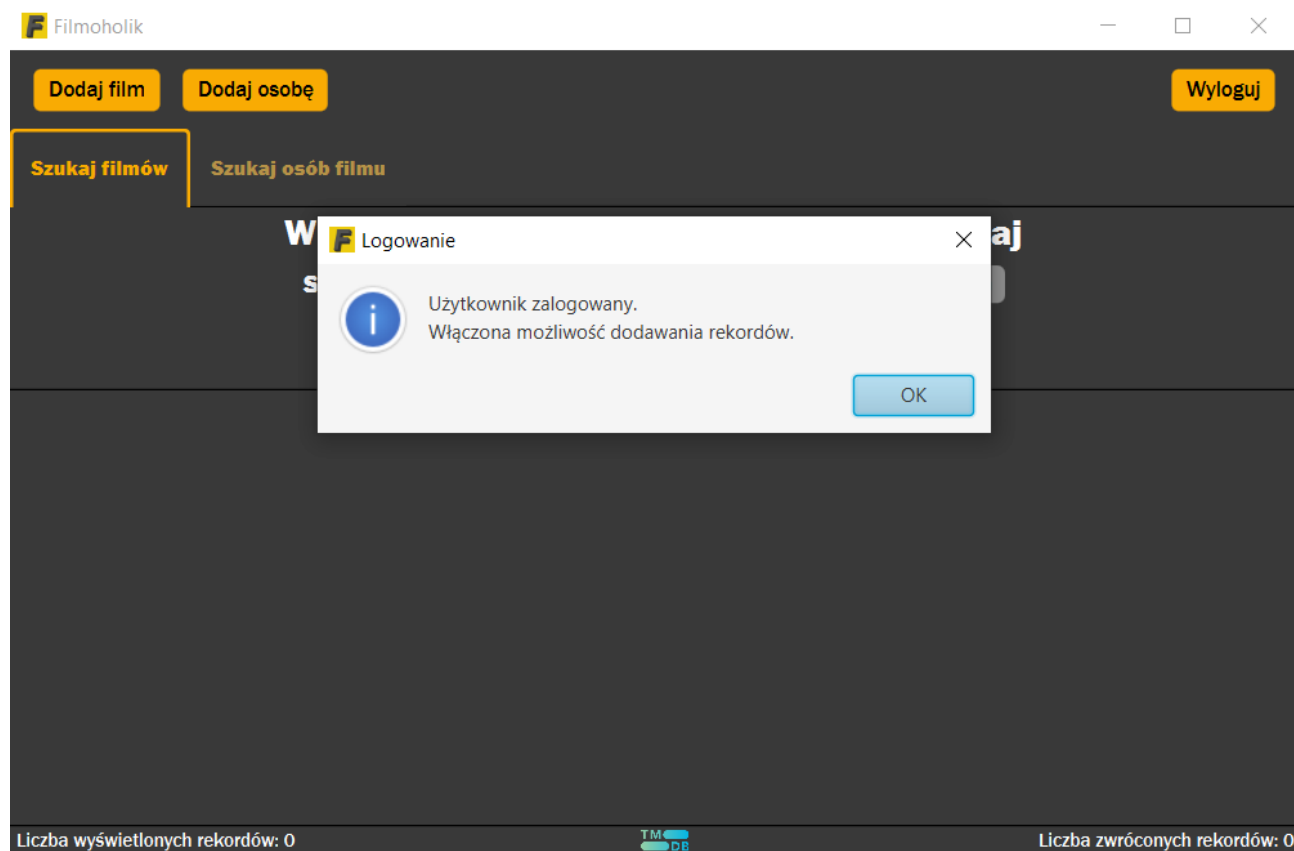
Natomiast na kliknięcie przycisków „Anuluj”, w obu przypadkach(formularz z rejestracją i logowaniem) aplikacja powróci do ekranu głównego.

W przypadku gdy niepoprawne jest hasło bądź nazwa użytkownika – pojawi się odpowiedni komunikat:



The screenshot shows a web browser window titled "Filmoholik". The main heading is "Logowanie". Below it, there are two input fields: "Nazwa użytkownika:" with the text "admin" and "Hasło:" with two dots. A red error message is displayed below the password field: "Błędna nazwa użytkownika lub hasło. Spróbuj ponownie." At the bottom, there are two buttons: "Zaloguj ✓" and "Anuluj ✕".

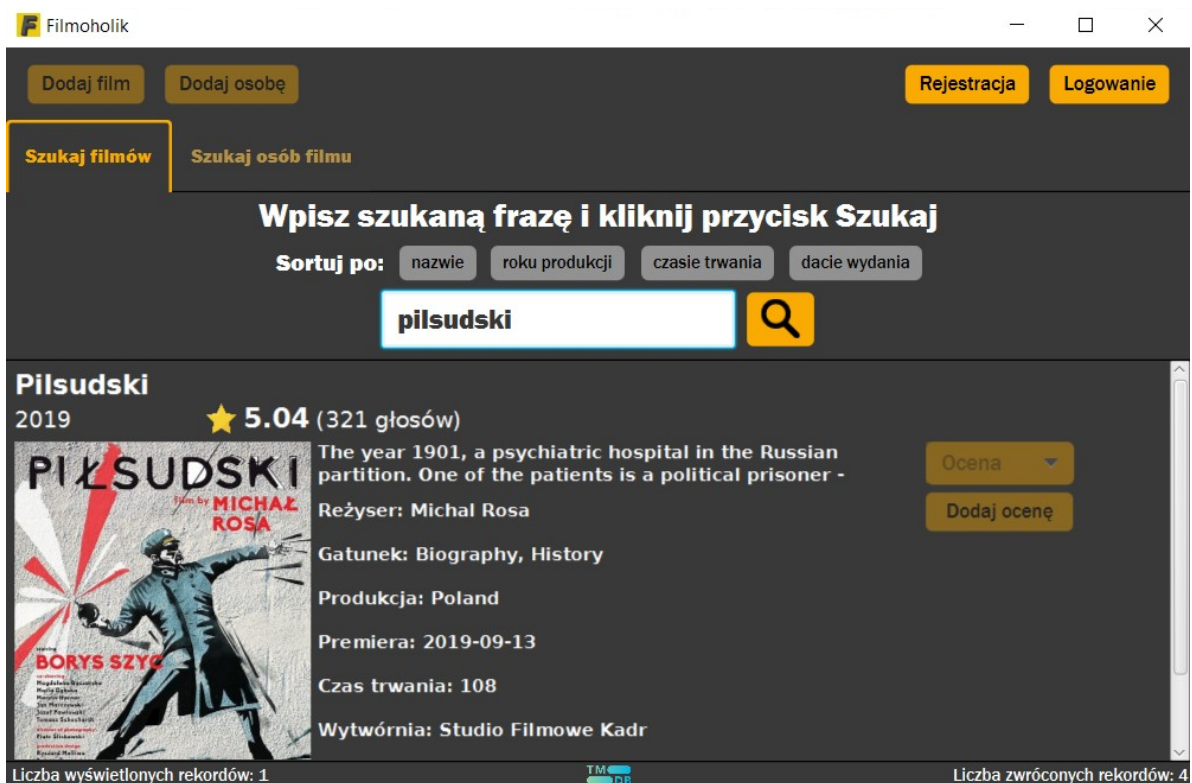
Natomiast kiedy hasło i nazwa się zgadzają, pojawia się alert informacyjny:



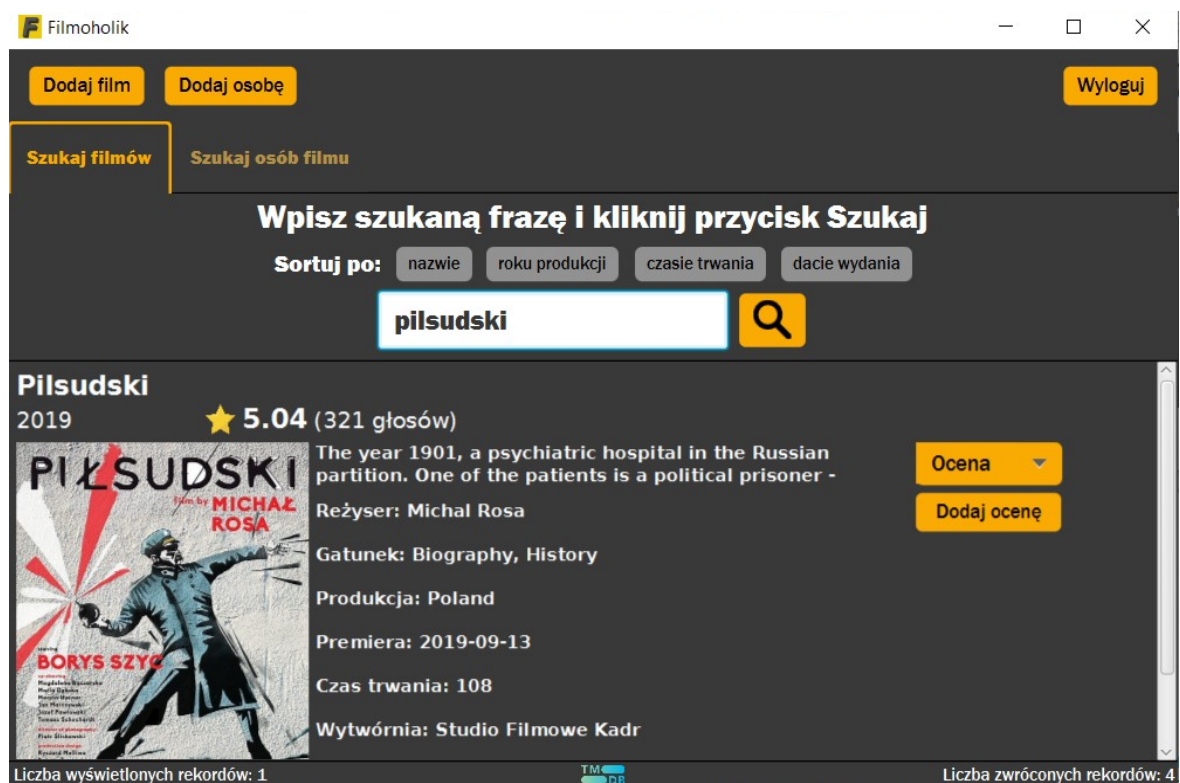
The screenshot shows the main interface of the "Filmoholik" application. At the top, there are buttons for "Dodaj film", "Dodaj osobę", and "Wyloguj". Below these are search tabs: "Szukaj filmów" and "Szukaj osób filmu". A modal dialog box titled "Logowanie" is open in the center, displaying an information icon and the text: "Użytkownik zalogowany. Włączona możliwość dodawania rekordów." with an "OK" button. At the bottom of the application, there is a footer with the text "Liczba wyświetlonych rekordów: 0" on the left, a "TM DB" logo in the center, and "Liczba zwróconych rekordów: 0" on the right.

Dla zalogowanego użytkownika udostępnione zostają przyciski widoczne w lewym górnym rogu powyższego zdjęcia oraz przyciski odpowiadające za dodanie oceny, występujące przy wyświetlonych produkcjach – rysunek poglądowy:

a) przed zalogowaniem:



b) po zalogowaniu:



Na kliknięcie przycisku „Wyloguj” użytkownik zostaje przekierowany do ekranu głównego, a przyciski na powrót zostają zablokowane.

Na ekranie głównym użytkownik ma możliwość swobodnego przechodzenia w zakładki „Szukaj filmów” oraz „Szukaj osób filmu” z odpowiednimi wyszukiwarkami i przyciskami sortowania dla danej zakładki.

Ekran po przejściu w zakładkę „Szukaj osób filmu”:

The screenshot shows the 'Szukaj osób filmu' (Search actors) tab selected in the Filmoholik application. The interface includes a header with navigation links 'Dodaj film' and 'Dodaj osobę', and user options 'Rejestracja' and 'Logowanie'. The main search area prompts the user to 'Wpisz szukaną frazę i kliknij przycisk Szukaj' (Enter the search phrase and click the Search button). Below this, there are sorting options 'Sortuj po:' with buttons for 'imieniu i nazwisku', 'zawodzie', and 'dacie narodzin'. A search input field contains the placeholder text 'Szukaj w bazie osób...' and is accompanied by a magnifying glass icon. The bottom status bar shows 'Liczba wyświetlonych rekordów: 0' (Number of displayed records: 0) and 'Liczba zwróconych rekordów: 0' (Number of returned records: 0).

Wpisywanie w wyszukiwarce danej zakładki i kliknięcie Enter bądź ikonki z lupą zwróci wyniki dla szukanej frazy. Jeżeli wynik nie zostanie odszukany aplikacji poinformuje o tym wyświetlając odpowiedni napis.

Pomyślne wyszukanie:

Dodaj film

Dodaj osobę

Rejestracja

Logowanie

Szukaj filmów

Szukaj osób filmu

Wpisz szukaną frazę i kliknij przycisk Szukaj

Sortuj po: imieniu i nazwisku zawodzie dacie narodzin

magda



Magdalena Boczarska



Data narodzin: 1978-12-12

Data śmierci: Brak

Biografia: Magdalena Boczarska is a Polish film and television actress. She was born December 1978. Her debut was at the New Theatre in Lodzi, she was playing the title directed by Lukasz Kos. Since 2003 she works on the boards of the National Theatre Berlin and Teatro Tatro Slovakia. Since 2005 Magdalena plays in popular television s. Later she acted in the lead roles in popular films in Europe.

Zawód: actress

Liczba wyświetlonych rekordów: 3



Liczba zwróconych rekordów: 3

Dodaj film

Dodaj osobę

Rejestracja

Logowanie

Szukaj filmów

Szukaj osób filmu

Wpisz szukaną frazę i kliknij przycisk Szukaj

Sortuj po: nazwie roku produkcji czasie trwania dacie wydania

pilsudski



Pilsudski

2019

★ 5.04 (321 głosów)



The year 1901, a psychiatric hospital in the Russian partition. One of the patients is a political prisoner -

Reżyser: Michał Rosa

Gatunek: Biography, History

Produkcja: Poland

Premiera: 2019-09-13

Czas trwania: 108

Wytwórnia: Studio Filmowe Kadr

Język: Polish

Obsada: [Borys Szyc] [Magdalena Boczarska] [Maria Debska]

Ocena

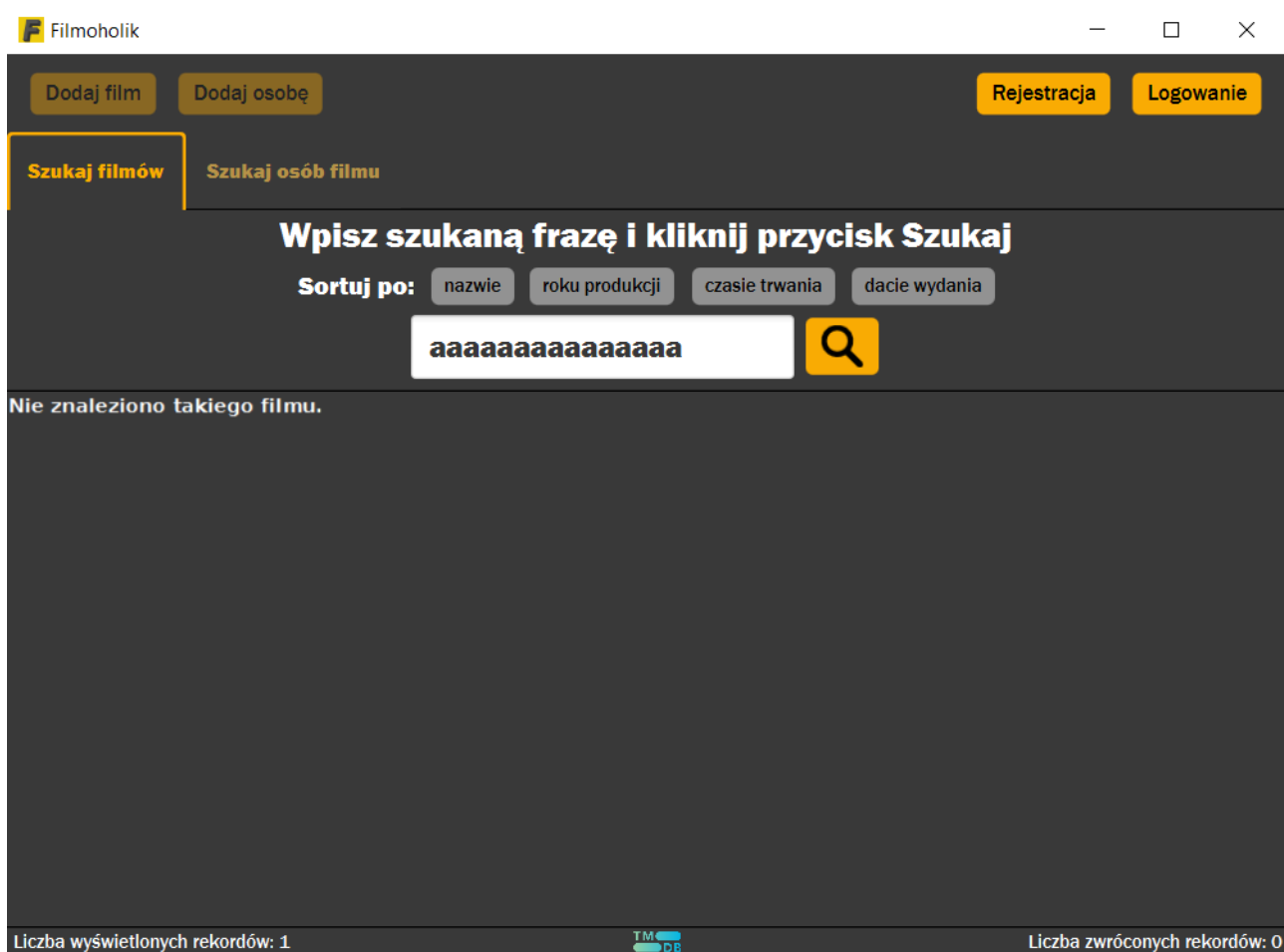
Dodaj ocenę

Liczba wyświetlonych rekordów: 1



Liczba zwróconych rekordów: 4

Natomiast w przypadku braku szukanego wyniku:



W lewym dolnym rogu aplikacji znajduje się etykieta z liczbą wyświetlonych rekordów, natomiast w prawym dolnym rogu etykieta z liczbą zwróconych rekordów. Z racji, że zwykle liczba zwróconych może przeważać nad liczbą wyświetlonych oraz liczba zwróconych jest zbyt duża dla aplikacji do przetworzenia w krótkim czasie, została ustawiona maksymalna liczba wyświetlanych rekordów, która wynosi 5.

Wyniki można sortować używając przycisków dostępnych na głównym ekranie. Z uwagi na fakt wspomniany w poprzednim paragrafie wyniki sortowania mogą być nieintuicyjne, gdyż sortowanie odbywa się po wszystkich rekordach nawet tych niewyświetlonych niemniej jednak działa ono poprawnie.

Przykład sortowania:

- przed:

Filmoholik

Dodaj film

Dodaj osobę

Rejestracja

Logowanie

Szukaj filmów

Szukaj osób filmu

Wpisz szukaną frazę i kliknij przycisk Szukaj

Sortuj po:

nazwie

roku produkcji

czasie trwania

datcie wydania

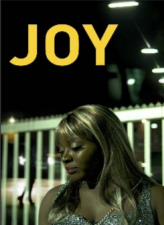
joy

Q

Joy

2018

★ 6.70 (1016 głosów)



Joy, a young Nigerian woman caught in the vicious cycle of sex trafficking, is instructed by her exploiter Madame to supervise Precious, a teenage girl who is not ready to accept her fate.

Reżyser: Sudabeh Mortezaei

Gatunek: Drama

Produkcja: Austria

Premiera: 2019-01-18

Czas trwania: 99

Wytwórnia: FreibuterFilm

Język: English, German

Obsada: [Anwulika Alphonsus] [Mariam Sanusi] [Angela Ekeleme] [Gift Igweh]


Ocena

Dodaj ocenę

Joyeuse retraite!

2019

★ 5.00 (258 głosów)



Marilou and Philippe prepare for their future retirement in Portugal. But their daughter separates and a whole procession of solicitations falls on them.

Reżyser: Fabrice Bracq

Gatunek: Comedy

Produkcja: France

Premiera: 2019-11-20

Ocena

Dodaj ocenę

Liczba wyświetlonych rekordów: 3

1/16 600 111

Liczba zwróconych rekordów: 14

- po sortowaniu alfabetycznym:

Filmoholik

Dodaj film

Dodaj osobę

Rejestracja

Logowanie

Szukaj filmów

Szukaj osób filmu

Wpisz szukaną frazę i kliknij przycisk Szukaj

Sortuj po:

nazwie

roku produkcji

czasie trwania

datcie wydania


joy

Q

Boyz in the Wood

2019

★ 6.40 (1589 głosów)



An anarchic, hip-hop inspired comedy that follows four city boys on a wilderness trek as they try to escape a mysterious huntsman.

Reżyser: Ninian Doff

Gatunek: Action, Comedy, Horror

Produkcja: UK

Premiera: 2020-08-28

Czas trwania: 87

Wytwórnia: Highland Midgie

Język: English

Obsada: [Kevin Guthrie] [James Cosmo] [Eddie Izzard]


Ocena

Dodaj ocenę

Joy

2018

★ 6.70 (1016 głosów)



Joy, a young Nigerian woman caught in the vicious cycle of sex trafficking, is instructed by her exploiter Madame to supervise Precious, a teenage girl who is not ready to accept her fate.

Reżyser: Sudabeh Mortezaei

Gatunek: Drama

Produkcja: Austria

Premiera: 2019-01-18

Ocena

Dodaj ocenę

Liczba wyświetlonych rekordów: 3

1/16 600 111

Liczba zwróconych rekordów: 14

Jak już wspomniano wcześniej – dla zalogowanych użytkowników możliwe jest dodawanie rekordów – nowych filmów oraz nowych osób, formularze dla tych opcji widoczne są poniżej:

Formularz z filmem:

 Filmoholik

— □ ×

Dodaj nową produkcję filmową

Tytuł:

Rok produkcji (4 cyfry):

Data wydania(YYYY-MM-dd):

Czas trwania(w minutach):

Opis:

Gatunek:

Kraj:

Wytwornia:

Język:

* Wszystkie pola są obowiązkowe

Zatwierdź ✓

Anuluj ✕

Formularz z osobą:

 Filmoholik

— □ ×

Dodaj nową osobę

Imię i nazwisko:

Data narodzin(opcjonalnie):

YYYY-MM-dd

Data śmierci(opcjonalnie):

YYYY-MM-dd

Krótką biografią(opcjonalnie):

Bio

Filmy, w którym pracował/a:
(oddzielone przecinkiem)

Film

Zawód(zawody):

☐ aktor

☐ reżyser

☐ producent

☐ aktorka

☐ kamerzysta

☐ archiwista

☐ scenarzysta

☐ scenograf

☐ dźwiękowiec

☐ kompozytor

☐ edytor

☐ on sam

Zatwierdź ✓

Anuluj ✕

Kliknięcie przycisków „Zatwierdź” w obu przypadkach dodaje daną produkcję / osobę do odpowiednich tabel, pod warunkiem że wpisane dane zostały wprowadzone poprawnie.

Przykładowy komunikat o niepoprawności danych:

Filmoholik

Dodaj nową produkcję filmową

Tytuł:

1234

Rok produkcji (4 cyfry):

12341

Błędny format

Data wydania(YYYY-MM-dd):

1234-12-121

Błędny format

Czas trwania(w minutach):

123

Opis:

aa

Gatunek:

aa

Kraj:

aa

Wytwornia:

aaaa

Jezyk:

aa

* Wszystkie pola są obowiązkowe

Zatwierdź ✓

Anuluj ✕

Przy poprawnym wypełnieniu pól, rekord zostaje wstawiony i zostaje wyświetlony komunikat o dodaniu rekordu:

Filmoholik

Dodaj film

Dodaj osobę

Wyloguj

Szukaj filmów

Szukaj osób filmu

W

S

aj

Formularz nowej produkcji

Produkcja została dodana.

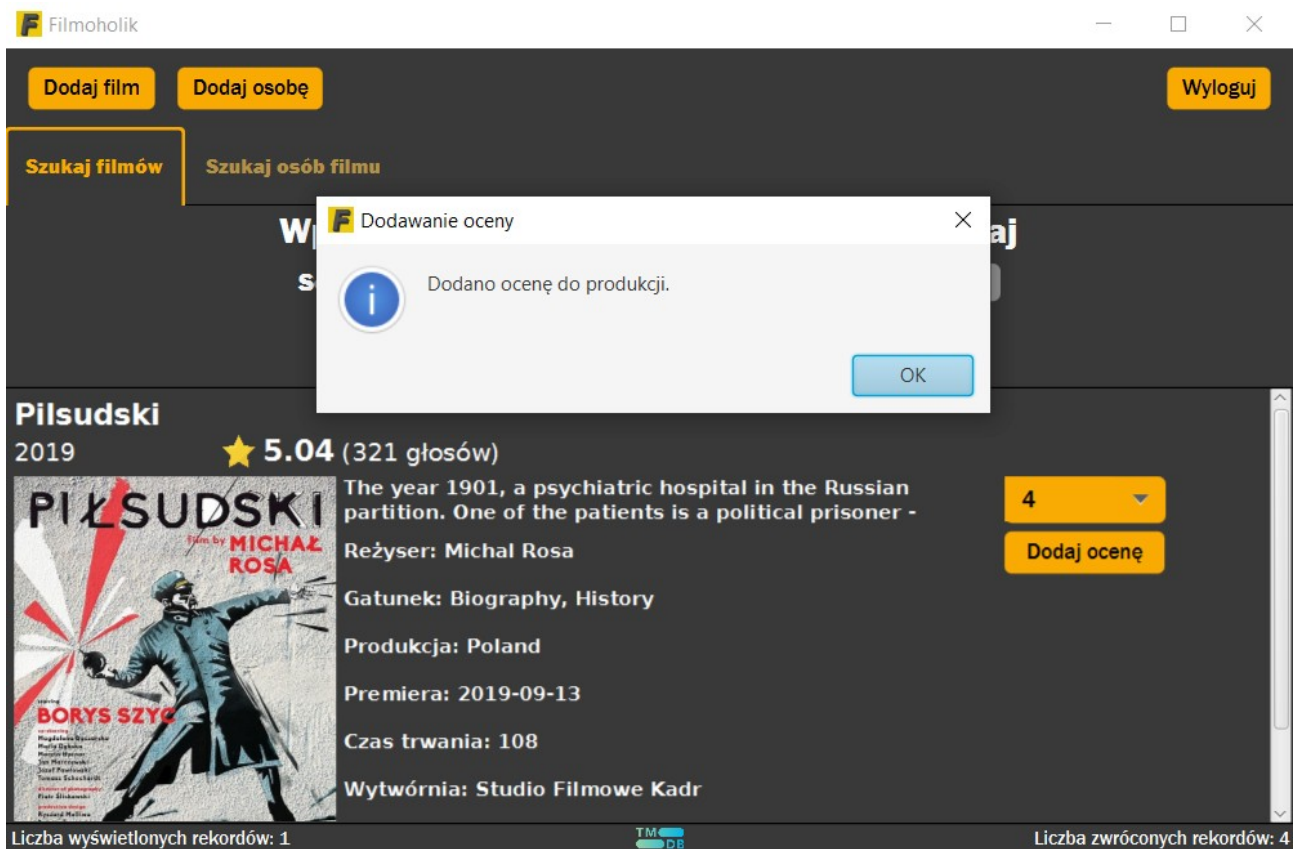
OK

Liczba wyświetlonych rekordów: 0

TM DB

Liczba zwróconych rekordów: 0

Natomiast dodawanie oceny polega na wybraniu odpowiedniego elementu z listy ComboBoxa o tytule „Ocena” , czyli oceny dla produkcji i kliknięciu przycisku „Dodaj ocenę”:



5.3 Opracowanie dokumentacji technicznej

Dokumentacja techniczna została wygenerowana automatycznie z poziomu aplikacji IntelliJ przy użyciu narzędzia Javadoc. Znajduje się w folderze „JavaDoc” (główna strona: ***index.html***).