

POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

PROJEKT Z BAZ DANYCH

Temat projektu

Baza danych kina

Termin zajęć: Wtorek, 13:15–15:00

AUTORZY:

Karol Jaskółka 241306

Mateusz Krawczak 241318

PROWADZĄCY ZAJĘCIA:

dr inż. Roman Ptak, W4/K9

Wrocław, 21.01.2020 r.

Spis treści:

1. Wstęp	4
1.1. Cel projektu.....	4
1.2. Zakres projektu	4
2. Analiza wymagań	4
2.1. Opis działania i schemat logiczny systemu	4
2.2. Wymagania funkcjonalne	4
2.3. Wymagania niefunkcjonalne	5
2.3.1. Wykorzystywane technologie i narzędzia.....	5
2.3.2. Wymagania dotyczące rozmiaru bazy danych	5
2.3.3. Wymagania dotyczące bezpieczeństwa systemu	5
3. Projekt systemu	6
3.1. Projekt bazy danych	6
3.1.1. Analiza rzeczywistości i uproszczony model konceptualny	6
3.1.2. Normalizacja.....	6
3.1.3. Model fizyczny i ograniczenia integralności danych.....	7
3.1.4. Inne elementy schematu – mechanizmy przetwarzania danych	7
3.1.5. Projekt mechanizmów bezpieczeństwa na poziomie bazy danych	10
3.2. Projekt aplikacji użytkownika.....	11
3.2.1. Architektura aplikacji i diagramy projektowe	11
3.2.2. Interfejs graficzny i struktura menu	12
3.2.3. Diagram przypadków użycia	13
3.2.4. Metoda podłączania do bazy danych – integracja z bazą danych.....	14
3.2.5. Projekt zabezpieczeń na poziomie aplikacji	14
4. Implementacja systemu baz danych	14
4.1. Tworzenie tabel i definiowanie ograniczeń	14
4.2. Implementacja mechanizmów przetwarzania danych	15
4.3. Implementacja uprawnień i innych zabezpieczeń	16
4.4. Testowanie bazy danych na przykładowych danych.....	17
5. Implementacja i testy aplikacji.....	20
5.1. Instalacja i konfigurowanie systemu	20
5.2. Instrukcja użytkowania aplikacji oraz testowanie funkcji systemu.....	23
5.3. Omówienie wybranych rozwiązań programistycznych	30

5.3.1. Implementacja interfejsu dostępu do bazy danych	30
5.3.2. Implementacja wybranych funkcjonalności systemu	31
5.3.3. Implementacja mechanizmów bezpieczeństwa.....	33
6. Podsumowanie i wnioski	33

1. Wstęp

1.1. Cel projektu

Celem projektu jest stworzenie systemu bazodanowego do zarządzania kinem. System będzie realizował różne funkcjonalności z podziałem na role. Administrator systemu będzie odpowiedzialny za zarządzanie bazą filmów oraz zarządzaniem poszczególnych uprawnień użytkowników. Pracownicy kina będą odpowiedzialni za dystrybucję biletów i rezerwacji. Od klienta będzie wymagane założenie konta w celu korzystania z czynności takich jak rezerwacja biletów czy tworzenie opinii na temat filmów.

1.2. Zakres projektu

Zakres projektu będzie obejmował analizę wymagań oraz kolejno projekt bazy danych na ich podstawie. Następnym etapem będzie implementacja systemu bazodanowego przy użyciu wybranego systemu do zarządzania bazą danych oraz implementacja aplikacji dostępowej do jego obsługi przy użyciu ORM.

2. Analiza wymagań

2.1. Opis działania i schemat logiczny systemu

Otoczenie systemu:

Nasze kino będzie posiadać trzy sale kinowe. Dwie mniejsze o pojemności 20 osób oraz jedną dużą dla 40 widzów. Dziennie w jednej sali będziemy puszczać maksymalnie 3 filmy. Kino będzie działać od godziny 12 do 23. W ciągu każdego miesiąca planowane jest około 10 światowych premier. W kinie będzie zatrudnionych 10 pracowników. W ciągu roku będzie miało miejsce około 3000 seansów. Bilet ulgowy będzie kosztował 15 zł, natomiast normalny 20 zł. Szacujemy, że średnio na jednym seansie będzie 25 osób, z czego stosunek biletów normalnych do ulgowych będzie wynosił 60:40 co daje 450 zł przychodu z jednego seansu. Przychód z całego roku będzie oscylował w granicach 1 350 000 zł. Odejmując koszty eksploatacyjne(prąd, awarie, filmy), płace pracowników i filmów roczne zyski kina szacujemy w granicach 900 000 zł.

2.2. Wymagania funkcjonalne

Administrator:

- Możliwość dodawania nowych filmów do bazy (tytuł, rok, reżyser, gatunek, ocena, opis, kategoria wiekowa).
- Możliwość edytowania i usuwania filmów z bazy.
- Usuwanie niestosownych komentarzy.
- Blokowanie i edytowanie kont użytkowników.
- Zarządzanie repertuarem seansów (data, godzina, sala, technologia).

Pracownik:

- Rezerwacja i sprzedaż biletów na seans.
- Usuwanie rezerwacji w przypadku nieobecności klienta.

Klient:

- Rejestracja w systemie (login, e-mail, hasło, dane osobowe, data urodzenia).
- Logowanie w systemie.
- Edycja danych osobowych.
- Możliwość rezerwacji biletów na seans.
- Możliwość kupienia biletów na seans.
- Wyszukiwanie seansów według wybranych kryteriów.
- Ocenianie filmów.
- Dodawanie komentarzy na temat filmów.
- Wyświetlanie opinii o filmach.
- Zgłaszanie skarg dotyczących pracowników oraz kina.

2.3. Wymagania niefunkcjonalne

2.3.1. Wykorzystywane technologie i narzędzia

- Do utworzenia bazy danych wykorzystany zostanie system Microsoft SQL Server zarządzany przez zintegrowane środowisko SQL Server Management Studio.
- Dostęp do graficznego interfejsu przez aplikację desktopową.
- Aplikacja desktopowa napisana w języku C# przy użyciu Windows Forms.
- Jako ORM użyjemy Entity Framework dla ADO.NET

2.3.2. Wymagania dotyczące rozmiaru bazy danych

- Przechowywanie 3000 seansów oraz około 75000 biletów rocznie.
- Dodawanie do bazy 120 nowych filmów każdego roku.

2.3.3. Wymagania dotyczące bezpieczeństwa systemu

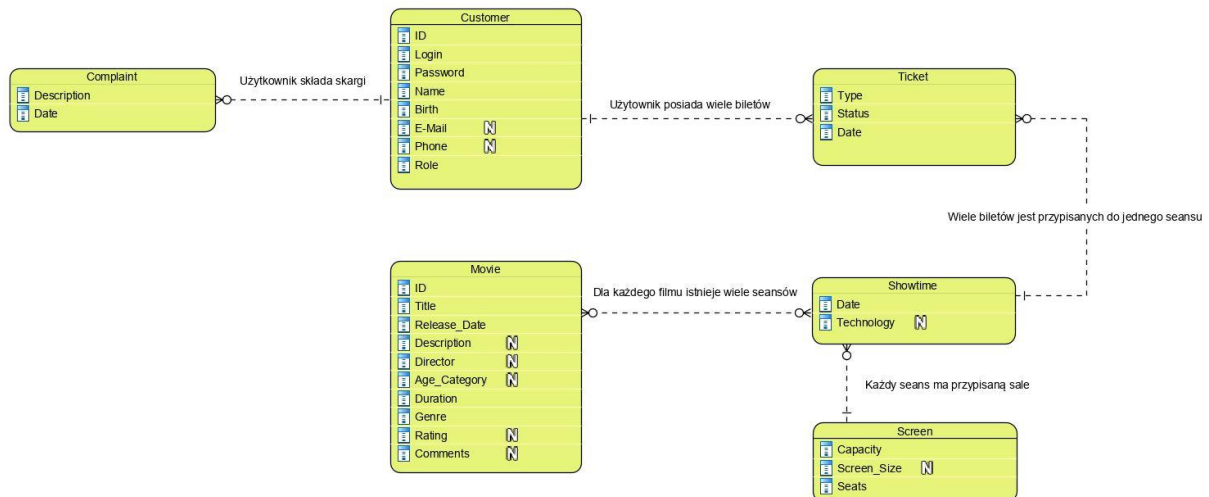
- Uwierzytelnianie danych logowania.
- Podział użytkowników na role (różne uprawnienia).
- Tylko administrator posiada możliwość zmiany uprawnień użytkowników.
- Zarządzanie filmami i seansami tylko przez uprawnione osoby.
- Ocenianie i dodawanie komentarzy tylko przez zalogowanych użytkowników.

3. Projekt systemu

Projekt i struktury bazy danych, mechanizmów zapewniania poprawności przechowywanych informacji, oraz kontroli dostępu do danych.

3.1. Projekt bazy danych

3.1.1. Analiza rzeczywistości i uproszczony model konceptualny



3.1.2. Normalizacja

- 1 Postać Normalna

Schemat relacji znajduje się w pierwszej postaci normalizacyjnej jeżeli wartości atrybutów są atomowe (niepodzielne)

Normalizacja :

- Atrybut Name w tabeli User znormalizowany do postaci First_Name i Last_Name
- Atrybut Type w tabeli Ticket opisujący typ biletu oraz wiążącą się z nim cenę znormalizowany do osobnej postaci Name oraz Price
- Atrybut Seats w tabeli Screen zawierający listę miejsc znajdujących się na sali nie spełnia 1 postaci normalnej więc wymagana normalizacji poprzez rozbicie na atomowe wartości co wiąże się z utworzeniem nowej encji

- 2 i 3 Postać Normalna

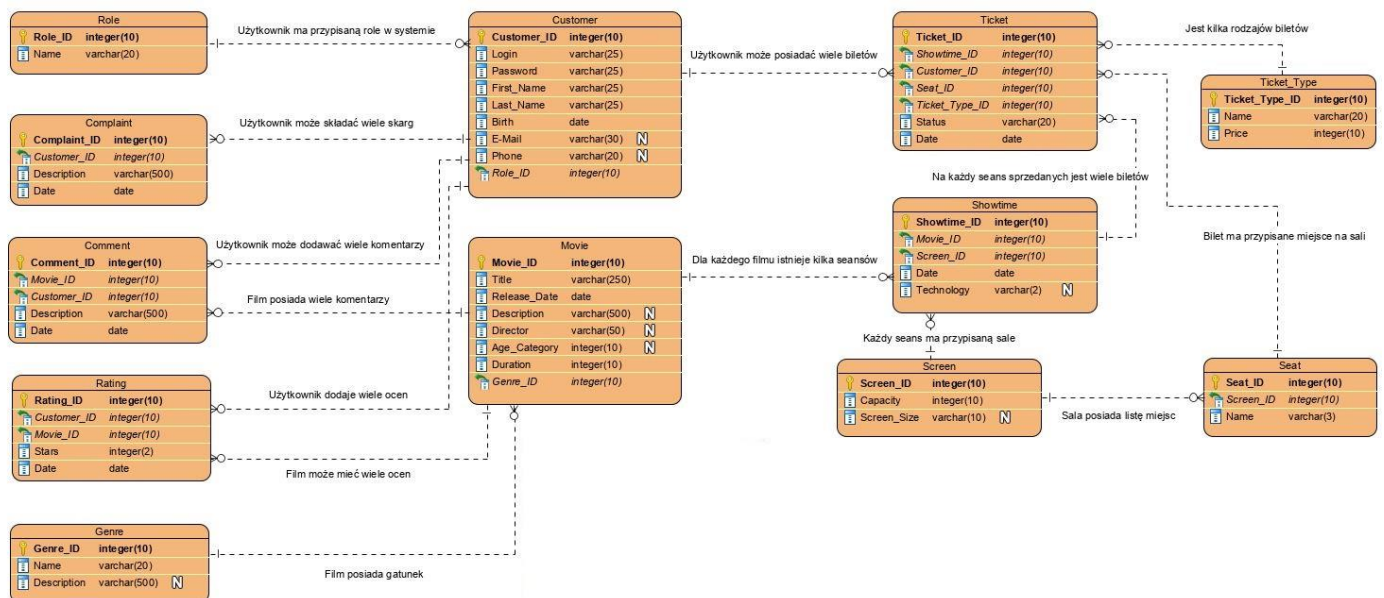
2 - W relacji nie mogą istnieć atrybuty wtórne, które są częściowo zależne od klucza.

3 - Relacja jest w trzeciej postaci normalnej tylko wtedy, gdy jest w drugiej postaci normalnej i każdy atrybut wtórny jest tylko bezpośrednio zależny od klucza głównego.

- Atrybuty Name i Price opisują jego pewien typ jako jedną całość więc zostaje dla nich utworzona nowa encja Ticket_Type

- Wydzielenie listy miejsc z encji Screen do nowej encji o nazwie Seat
- Z komentarzy do filmu zostaje utworzona osobna tabela Comment która będzie zawierać, która będzie powiązana przez klucze obce z danym filmem i użytkownikiem, który dodał komentarz
- Oceny do filmu także będą w osobnej tabeli o nazwie Rating gdyż działają na podobnej zasadzie jak komentarze
- W celu przechowywania nazwa i opisu gatunku filmy, encja Movie będzie zawierać klucz obcy do nowo utworzonej encji Genre

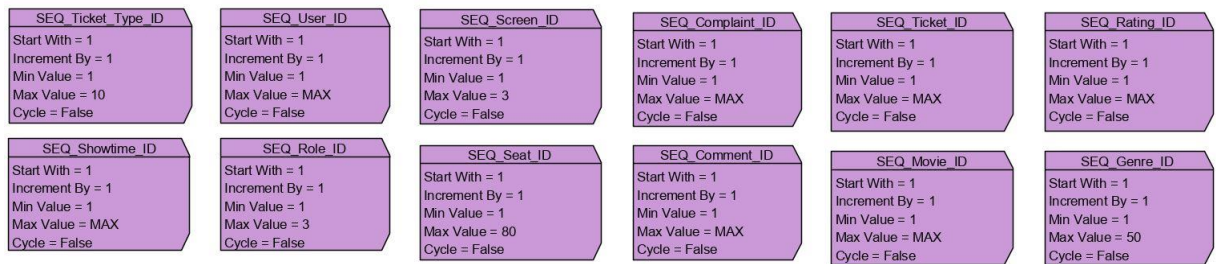
3.1.3. Model fizyczny i ograniczenia integralności danych



3.1.4. Inne elementy schematu – mechanizmy przetwarzania danych

- Widoki
 - Showtime_View – widok zawierający informacje na temat seansu o takich elementach jak : film (tytuł, czas trwania, kategoria wiekowa, reżyser, opis, gatunek), numer sali, data seansu oraz technologia projekcji filmu
 - Ticket_View – widok zawierający informację na temat biletu takich jak : imię i nazwisko posiadacza, tytuł filmu, data seansu, cena biletu i data jego zakupu, numer sali oraz miejsce na sali.
- Sekwencje

Dla każdej encji zostanie przypisana sekwencja do automatycznego przypisania kolejnych wartości ID według poniższego rysunku :



- Indeksy

W celu szybszego dostępu do danych zostaną wykorzystane następujące indeksy :

- Indeks w tabeli Ticket po atrybucie Customer_ID
- Indeks w tabeli Movie po atrybucie Title
- Indeks w tabeli Comment po atrybucie Date malejąco
- Indeks w tabeli Rating po atrybucie Movie_ID
- Indeks w tabeli Showtime po atrybucie Date malejąca
- Indeks w tabeli Customer po atrybucie Last_Name
- Indeks w tabeli Customer po atrybutach Login i Password
- Indeks w tabeli Movie po atrybucie Release_Date

- Triggery

- Rating_Triggers
 - rating_stars_ai – ustawia automatycznie ocenę wprowadzoną przez użytkownika na 1 w przypadku wystawienia oceny mniejszej od 1 i analogicznie w przypadku wystawienia oceny większej od 10 ustawia na 10 czyli maksymalna wartość;
- Showtime_Triggers
 - showtime_technology_ai – ustawia wartość na 2d w przypadku ustawienia wartości innej od 2d lub 3d;
- Movie_Triggers
 - movie_age_category_ai – jeśli zostanie wprowadzona wartość mniejsza od 3, zostanie ona ostawiona na 3 i analogicznie jeśli będzie większa od 18 wartość zostanie ustawiona na 18;

- movie_duration_ai – jeśli wartość trwania filmu będzie mniejsza od 1 to zostanie ustawiona na 1 - wartość minimalna;
- Ticket_Type_Triggers
 - ticket_type_price_ai – jeśli cena zostanie ustawiona na mniejszą od 15 automatycznie zostanie ustawiona na 15, tak samo w przypadku gdy cena będzie większa od 25;
- Procedury Składowane
 - Rating_procedures
 - sp_showCustomerRatings() – pokazuje wszystkie oceny wystawione przez użytkownika;
 - sp_showAverageRatingMovie() – pokazuje średnią ocen wystawionych przez użytkowników do konkretnego filmu;
 - Ticket_procedures
 - sp_showTickets() – pokazuje wszystkie bilety na dany seans
 - Seat_procedures
 - sp_showAvailableSeats() – pokazuje wszystkie wolne siedzenia na dany seans
 - Genre_Procedures
 - sp_showGenre() – pokazuje gatunek wybranego filmu
 - Comment_procedures
 - sp_showCommentsCustomer() – pokazuje wszystkie komentarze wystawione przez użytkownika;
 - sp_showCommentsMovie() – pokazuje wszystkie komentarze wystawione przez użytkowników do konkretnego filmu;
 - Complaint_procedures
 - sp_showComplaints() – pokazuje wszystkie skargi złożone przez użytkownika;
 - Customer_procedures
 - sp_showCustomers () – pokazuje wszystkich użytkowników;
 - sp_showCustomer () – pokazuje dane konkretnego użytkownika;

- Movie_procedures
 - sp_showMovies () – pokazuje wszystkie filmy;
 - sp_showMovie () – pokazuje dane konkretnego filmu;

3.1.5. Projekt mechanizmów bezpieczeństwa na poziomie bazy danych

- Granty

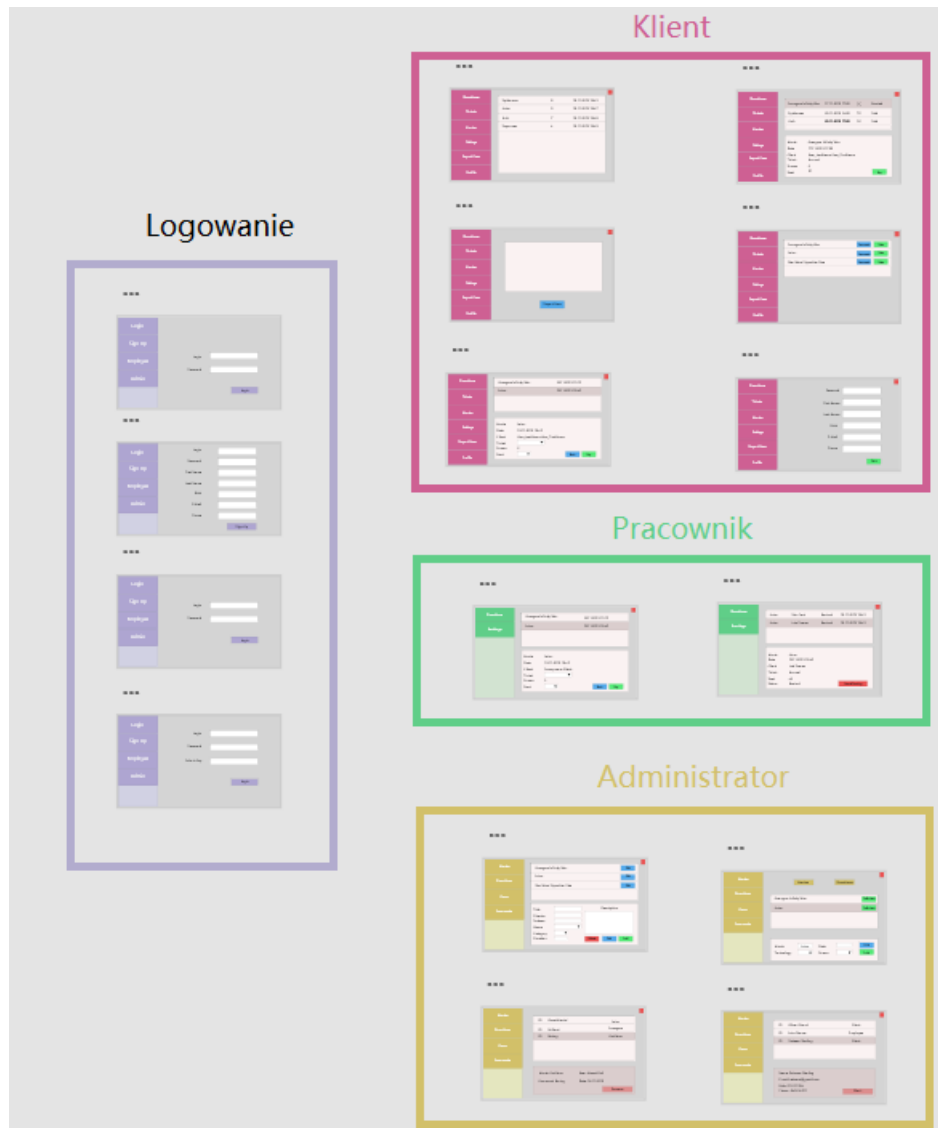
Nazwa tabeli	SELECT	INSERT	UPDATE	DELETE	CREATE
Role	Admin	Admin	Admin	Admin	Admin
Complaint	*	Admin, Klient	Admin, Klient	Admin	Admin
Comment	*	Admin, Klient	Admin, Klient	Admin, Klient	Admin
Rating	*	Admin, Klient	Admin, Klient	Admin, Klient	Admin
Genre	*	Admin	Admin	Admin	Admin
Customer	*	*	*	Admin	Admin
Movie	*	Admin	Admin	Admin	Admin
Ticket	*	*	*	Admin, Employee	Admin
Showtime	*	Admin	Admin	Admin	Admin
Screen	*	Admin	Admin	Admin	Admin
Seat	*	Admin	Admin	Admin	Admin
Ticket_type	*	Admin	Admin	Admin	Admin

* - wszyscy

3.2. Projekt aplikacji użytkownika

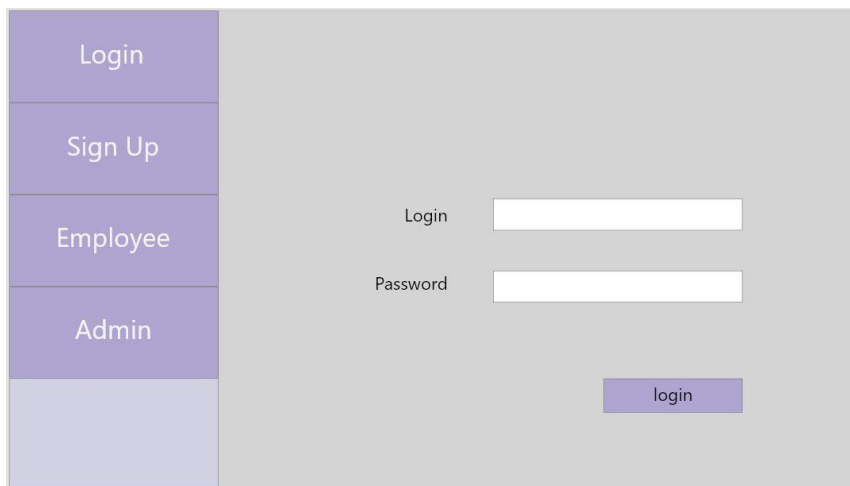
3.2.1. Architektura aplikacji i diagramy projektowe

- Aplikacja będzie składała się z okna logowania z rozróżnieniem dla trzech typów użytkowników oraz dodatkowym panelem do rejestracji w systemie.
- Klient, pracownik oraz administrator po zalogowaniu będzie miał dostęp do przeznaczonego dla niego okna aplikacji.



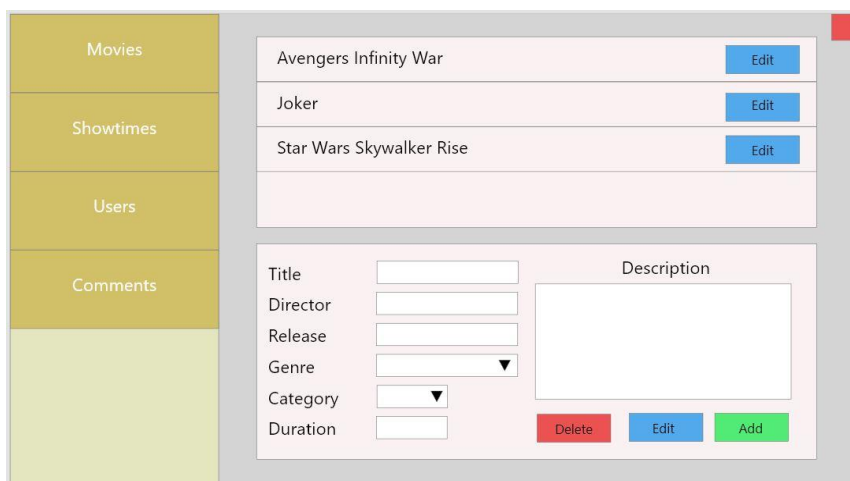
3.2.2. Interfejs graficzny i struktura menu

- Ekran startowy (logowania)



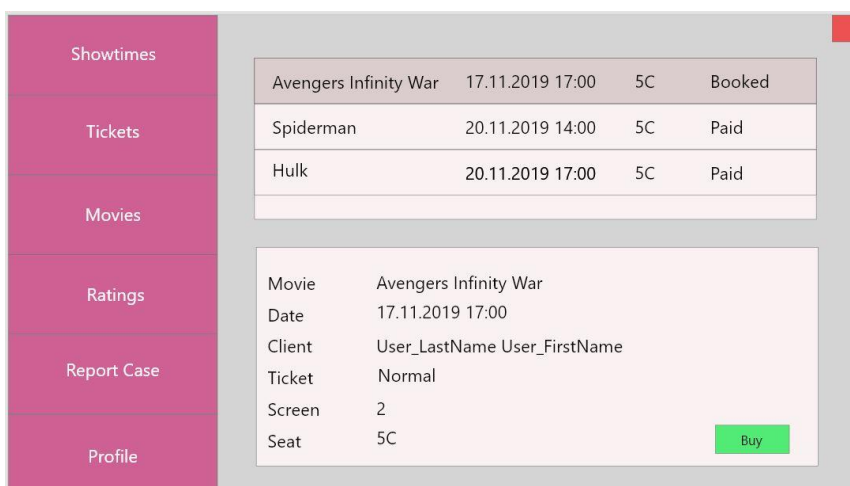
The login screen features a vertical sidebar on the left with five buttons: 'Login', 'Sign Up', 'Employee', 'Admin', and an empty light blue button. The main area has a light gray background with labels 'Login' and 'Password' next to white input fields. A purple 'login' button is positioned at the bottom right.

- Ekran administratora



The administrator screen has a sidebar with 'Movies', 'Showtimes', 'Users', 'Comments', and an empty light green button. The main area displays a list of movies: 'Avengers Infinity War', 'Joker', and 'Star Wars Skywalker Rise', each with an 'Edit' button. Below this is a form for adding or editing a movie, with fields for Title, Director, Release, Genre (dropdown), Category (dropdown), and Duration. A large text area for Description is also present. At the bottom right of the form are 'Delete', 'Edit', and 'Add' buttons.

- Ekran klienta



The client screen has a sidebar with 'Showtimes', 'Tickets', 'Movies', 'Ratings', 'Report Case', and 'Profile'. The main area shows a table of movie showtimes:

Movie	Date	Time	Screen	Status
Avengers Infinity War	17.11.2019	17:00	5C	Booked
Spiderman	20.11.2019	14:00	5C	Paid
Hulk	20.11.2019	17:00	5C	Paid

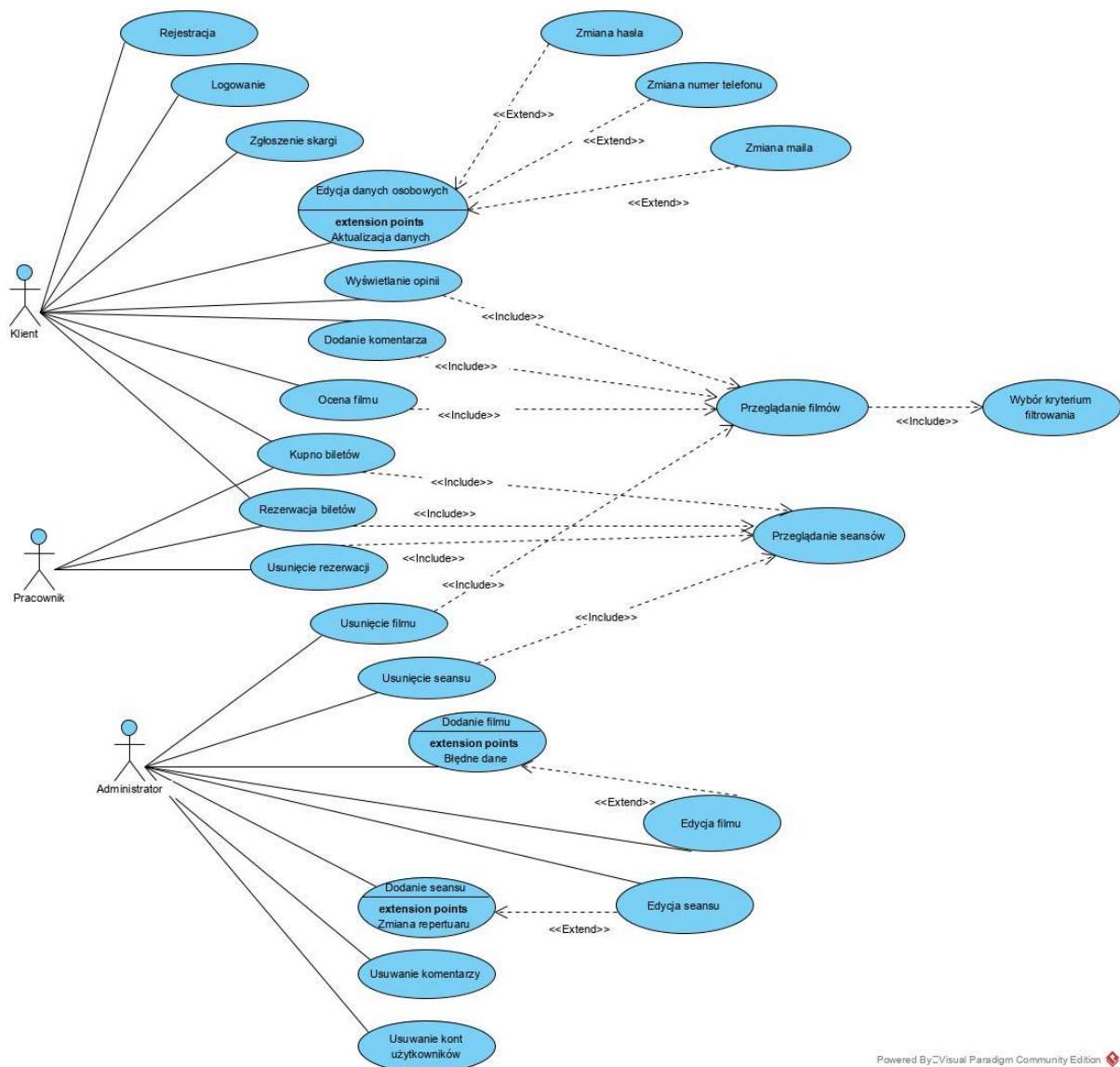
Below the table is a form for purchasing a ticket, with fields for Movie, Date, Client (User_LastName User_FirstName), Ticket, Screen, and Seat. A green 'Buy' button is at the bottom right.

- Ekran pracownika

Showtimes	
Avengers Infinity War	16.11.2019 15:15
Joker	18.11.2019 18:45

Movie	Joker
Date	18.11.2019 18:45
Client	Anonymous Client
Ticket	<input type="text"/>
Screen	3
Seat	<input type="text"/>
<input type="button" value="Book"/> <input type="button" value="Buy"/>	

3.2.3. Diagram przypadków użycia



3.2.4. Metoda podłączania do bazy danych – integracja z bazą danych

Aplikacja będzie podłączona do bazy danych za pomocą narzędzia mapowania obiektowo-relacyjnego (ORM) Entity Framework wersji 6.0+ dla ADO.NET.

3.2.5. Projekt zabezpieczeń na poziomie aplikacji

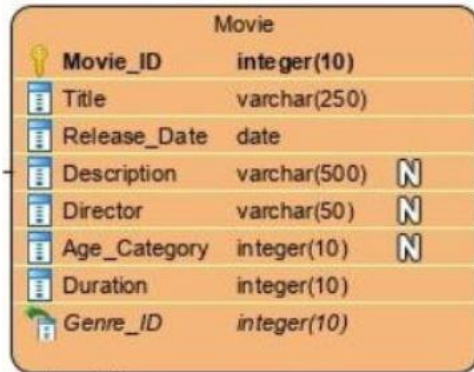
- Dostęp do aplikacji wymaga posiadania konta w systemie oraz poprawnego zalogowania się poprzez podanie prawidłowego loginu oraz hasła.
- Po zalogowaniu użytkownik (Klient, Pracownik lub Administrator) będzie miał możliwość wykonania akcji odpowiadających jego roli w systemie (trzy oddzielne GUI wynikającej z podziału na role).
- Wprowadzanie danych z poziomu aplikacji dla elementów z ograniczoną dziedziną będzie dostępne poprzez wybierane dostępnych opcji z list wyboru w celu uniknięcia wprowadzenia nieprawidłowych danych.

4. Implementacja systemu baz danych

Implementacja i testy bazy danych w systemie Microsoft SQL Server Management Studio

4.1. Tworzenie tabel i definiowanie ograniczeń

- Przykładowe utworzenie tabeli Movie zgodnie z diagramem.



```
CREATE TABLE Movie(  
    Movie_ID int PRIMARY KEY,  
    Title varchar(250) NOT NULL,  
    Release_Date date NOT NULL,  
    Description varchar(500),  
    Director varchar(50),  
    Age_Category int,  
    Duration int NOT NULL,  
    Genre_ID int  
);
```

- Dodanie odwołania do klucza obcego Genre_ID z tabeli Genre

```
ALTER TABLE Movie  
ADD CONSTRAINT FK_Movie_Genre  
FOREIGN KEY(Genre_ID)  
REFERENCES Genre(Genre_ID);
```

- Utworzenie sekwencji w celu przypisywania kolejnych wartości klucza głównego

```
CREATE SEQUENCE SEQ_Movie_ID
AS INT
START WITH 1
INCREMENT BY 1
MINVALUE 1
NO MAXVALUE
NO CYCLE;
```

4.2. Implementacja mechanizmów przetwarzania danych

- Widoki

```
CREATE VIEW SHOWTIME_VIEW AS
SELECT Movie.Title AS 'Movie', Movie.Duration, Movie.Age_Category AS 'Age Category', Movie.Director,
Movie.Description, Genre.Name AS 'Genre', Screen.Screen_ID AS 'Screen', s1.Technology,
(SELECT CONCAT(DATEPART(year, Date), '-', DATEPART(month, Date), '-', DATEPART(day, Date), ' ', DATEPART(hour, Date), ': ',
DATEPART(MINUTE, Date)))
FROM Showtime s2
WHERE s2.Showtime_ID=s1.Showtime_ID) AS 'Date'
FROM Showtime s1
JOIN Movie ON s1.Movie_ID = Movie.Movie_ID
JOIN Screen ON s1.Screen_ID = Screen.Screen_ID
JOIN Genre ON Movie.Genre_ID = Genre.Genre_ID;
```

- Triggery

```
CREATE TRIGGER rating_ai ON Rating
AFTER INSERT
AS
DECLARE @starsMin int ,@starsMax int
SELECT @starsMin=stars, @starsMax=stars FROM inserted
IF @starsMIN < 1
UPDATE Rating SET stars=1 WHERE stars<1
IF @starsMAX > 10
UPDATE Rating SET stars=10 WHERE stars>10
```

- Indeksy

```
-- INDEXES

CREATE NONCLUSTERED INDEX IX_MOVIE_TITLE ON Movie(Title);
CREATE NONCLUSTERED INDEX IX_TICKET_CUSTOMER ON Ticket(Customer_ID);
CREATE NONCLUSTERED INDEX IX_COMMENT_DATE ON Comment(Date DESC);
CREATE NONCLUSTERED INDEX IX_RATING_MOVIE ON Rating(Movie_ID);
CREATE NONCLUSTERED INDEX IX_SHOWTIME_DATE ON Showtime(Date DESC);
CREATE NONCLUSTERED INDEX IX_CUSTOMER_NAME ON Customer(Last_Name, First_Name);
CREATE NONCLUSTERED INDEX IX_MOVIE_RELEASE ON Movie(Release_Date);
CREATE NONCLUSTERED INDEX IX_CUSTOMER_AUTHENTICATION ON Customer(Login,Password);
```

- Procedury składowane

```
CREATE PROCEDURE sp_showAvailableSeats @Showtime_ID int
AS
-- SELECT ALL SEATS EXCEPT TAKEN ONES
SELECT Seat.Seat_ID, Seat.Name
FROM Seat
JOIN Screen ON Screen.Screen_ID = Seat.Screen_ID
JOIN Showtime ON Showtime.Screen_ID = Screen.Screen_ID
WHERE Showtime.Showtime_ID = @Showtime_ID
EXCEPT
SELECT Seat.Seat_ID, Seat.Name
FROM Ticket
JOIN Seat ON Ticket.Seat_ID = Seat.Seat_ID
JOIN Showtime ON Ticket.Showtime_ID = Showtime.Showtime_ID
WHERE Showtime.Showtime_ID = @Showtime_ID
GO
```

4.3. Implementacja uprawnień i innych zabezpieczeń

Przykład grantów dla klienta :

```
GRANT SELECT, INSERT, UPDATE, DELETE ON Comment TO CINEMA_CUSTOMER;
GRANT SELECT, INSERT, UPDATE, DELETE ON Rating TO CINEMA_CUSTOMER;
GRANT SELECT, INSERT, UPDATE ON Complaint TO CINEMA_CUSTOMER;
GRANT SELECT, INSERT, UPDATE ON Ticket TO CINEMA_CUSTOMER;
GRANT SELECT, INSERT, UPDATE ON Customer TO CINEMA_CUSTOMER;
GRANT SELECT ON Genre TO CINEMA_CUSTOMER;
GRANT SELECT ON Movie TO CINEMA_CUSTOMER;
GRANT SELECT ON Ticket_Type TO CINEMA_CUSTOMER;
GRANT SELECT ON Seat TO CINEMA_CUSTOMER;
GRANT SELECT ON Screen TO CINEMA_CUSTOMER;
GRANT SELECT ON Showtime TO CINEMA_CUSTOMER;
GRANT SELECT ON TICKET_VIEW TO CINEMA_CUSTOMER;
GRANT SELECT ON SHOWTIME_VIEW TO CINEMA_CUSTOMER;

GRANT EXEC ON sp_showCustomerRatings TO CINEMA_CUSTOMER;
GRANT EXEC ON sp_showMovieRatings TO CINEMA_CUSTOMER;
GRANT EXEC ON sp_showAverageRatingMovie TO CINEMA_CUSTOMER;
GRANT EXEC ON sp_showTickets TO CINEMA_CUSTOMER;
GRANT EXEC ON sp_showAvailableSeats TO CINEMA_CUSTOMER;
GRANT EXEC ON sp_showGenre TO CINEMA_CUSTOMER;
GRANT EXEC ON sp_showCommentsCustomer TO CINEMA_CUSTOMER;
GRANT EXEC ON sp_showCommentsMovie TO CINEMA_CUSTOMER;
GRANT EXEC ON sp_showComplaints TO CINEMA_CUSTOMER;
GRANT EXEC ON sp_showCustomer TO CINEMA_CUSTOMER;
GRANT EXEC ON sp_showCustomers TO CINEMA_CUSTOMER;
GRANT EXEC ON sp_showMovie TO CINEMA_CUSTOMER;
GRANT EXEC ON sp_showMovies TO CINEMA_CUSTOMER;

DENY DELETE ON Complaint TO CINEMA_CUSTOMER;
DENY DELETE ON Ticket TO CINEMA_CUSTOMER;
DENY DELETE ON Customer TO CINEMA_CUSTOMER;
DENY INSERT, UPDATE, DELETE ON Genre TO CINEMA_CUSTOMER;
DENY INSERT, UPDATE, DELETE ON Movie TO CINEMA_CUSTOMER;
DENY INSERT, UPDATE, DELETE ON Ticket_Type TO CINEMA_CUSTOMER;
DENY INSERT, UPDATE, DELETE ON Seat TO CINEMA_CUSTOMER;
DENY INSERT, UPDATE, DELETE ON Screen TO CINEMA_CUSTOMER;
DENY INSERT, UPDATE, DELETE ON Showtime TO CINEMA_CUSTOMER;
```


4.4. Testowanie bazy danych na przykładowych danych

Przykładowe wyniki zapytań :

- `SELECT * FROM SHOWTIME_VIEW WHERE Date LIKE '2020-1-22%' ORDER BY DATE ASC;`

	Movie	Duration	Age Category	Director	Description	Genre	Screen	Technology	Date
1	Whiplash	105	3	Damien Hazielle	NULL	Drama	1	2D	2020-1-22 16:30
2	Avengers	142	3	Josh Whedon	NULL	Sci-Fi	3	2D	2020-1-22 16:45
3	I am Legend	101	12	Francis Lawrence	NULL	Sci-Fi	1	2D	2020-1-22 18:30
4	Intouchables	108	3	Olivier Nakache	NULL	Comedy	3	2D	2020-1-22 19:30
5	Alien	117	16	Ridley Scott	NULL	Horror	2	2D	2020-1-22 20:45

- `EXEC sp_showMovies;`

	Title	Director	Genre	Release date	Age category
1	Joker	Todd Phillips	Crime	2019-10-04	16
2	Once Upon a Time... in Hollywood	Quentin Tarantino	Crime	2019-08-16	16
3	Frozen 2	Jennifer Lee	Animation	2019-11-22	3
4	Star Wars: The Rise of Skywalker	J.J. Abrams	Sci-Fi	2019-12-19	12
5	Parasite	Joon-Ho Bong	Drama	2019-09-20	12
6	Whiplash	Damien Hazielle	Drama	2015-01-02	3
7	Alien	Ridley Scott	Horror	1979-05-25	16
8	Silent Hill	Christophe Gans	Horror	2006-05-26	16
9	It	Andy Muschietti	Horror	2017-09-08	16
10	The Conjuring	James Wan	Horror	2013-07-26	16

- `EXEC sp_showCustomers;`

	Customer	Login	Birth date	Email	Phone
1	Jan Kowalski	Jan83	1983-04-22	kowal@gmail.com	615234004
2	Leo Messi	AtomicPulga	1989-10-26	lmessi@gmail.com	215488975
3	LeBron James	kinglj	1987-08-05	ljames@gmail.com	443258971
4	Eden Hazard	hazardeden	1992-07-11	ehazard@gmail.com	784525641
5	Cristiano Ronaldo	cr7	1988-08-01	cronaldo@gmail.com	756285341

- `EXEC sp_showAvailableSeats @Showtime_ID=4;`

	Seat_ID	Name
1	21	1A
2	23	3A
3	26	1B
4	29	4B
5	30	5B
6	31	1C
7	32	2C
8	33	3C
9	34	4C
10	35	5C
11	36	1D
12	37	2D
13	38	3D
14	39	4D
15	40	5D

- `SELECT * FROM TICKET_VIEW;`

	Client	Movie	Showtime	Ticket Price	Status	Seat	Screen_ID	Transaction
1	Jan Kowalski	Joker	2019-11-26 20:15	20	Paid	1A	3	2019-11-25 22:35
2	LeBron James	Frozen 2	2019-11-26 15:15	20	Booked	2A	2	2019-11-25 22:48
3	Eden Hazard	Frozen 2	2019-11-26 15:15	20	Paid	4A	2	2019-11-25 22:48
4	Cristiano Ronaldo	Frozen 2	2019-11-26 15:15	20	Paid	5A	2	2019-11-25 23:41
5	Christian Pulisic	Frozen 2	2019-11-26 15:15	20	Booked	2B	2	2019-11-25 19:22
6	Frank Lampard	Frozen 2	2019-11-26 15:15	20	Paid	3B	2	2019-11-25 17:36
7	Leo Messi	Star Wars: The Rise of Skywalker	2019-12-19 20:45	15	Paid	3B	3	2019-11-25 14:36

- `EXEC sp_showCommentsMovie @Movie_ID=1;`

	Title	Customer	Comment
1	Joker	Jan Kowalski	Amazing!

- Próba wykonania polecenia INSERT przez CINEMA_CUSTOMER

```
Msg 229, Level 14, State 5, Line 660
The INSERT permission was denied on the object 'Showtime', database 'Cinema', schema 'dbo'.
```

- Próba dodania wartości NULL w nieodpowiednim miejscu :

```
-- (ID, Customer_ID, Movie_ID, Stars, Date)
INSERT INTO Rating Values(NEXT VALUE FOR SEQ_RATING_ID, 2,NULL,10,'2020-01-06');
```

100 %

Messages

```
Msg 515, Level 16, State 2, Line 834
Cannot insert the value NULL into column 'Movie_ID', table 'Cinema.dbo.Rating'; column does not allow nulls. INSERT fails.
The statement has been terminated.
```

Testy Wydajnościowe dla tabeli Customer:

- 100 rekordów
 - INSERT

Elapsed time	00:00:00.2519638
--------------	------------------

- SELECT

Elapsed time	00:00:00.1609858
--------------	------------------

- 1000 rekordów

- INSERT

Elapsed time	00:00:01.8751436
--------------	------------------

- SELECT

Elapsed time	00:00:00.1959482
--------------	------------------

- 10 000 rekordów

- INSERT

Elapsed time	00:00:14.497895
--------------	-----------------

- SELECT

Elapsed time	00:00:00.3225027
--------------	------------------

- 100 000 rekordów

- INSERT

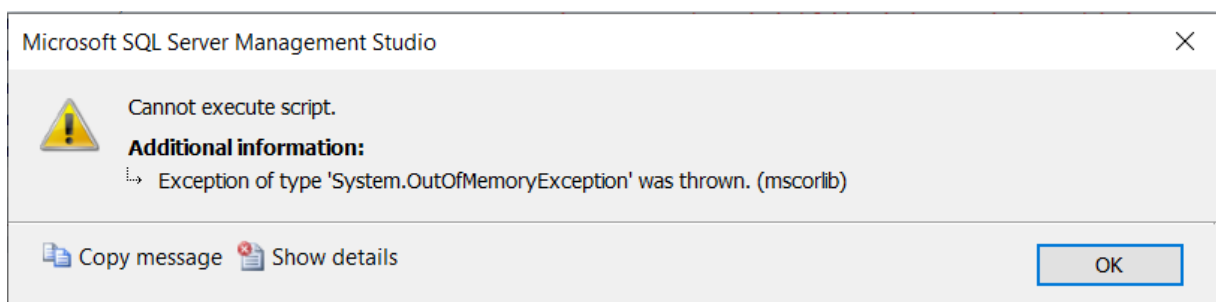
Elapsed time	00:05:53.7629617
--------------	------------------

- SELECT

Elapsed time	00:00:01.6653995
--------------	------------------

- 1 000 000 rekordów

- INSERT



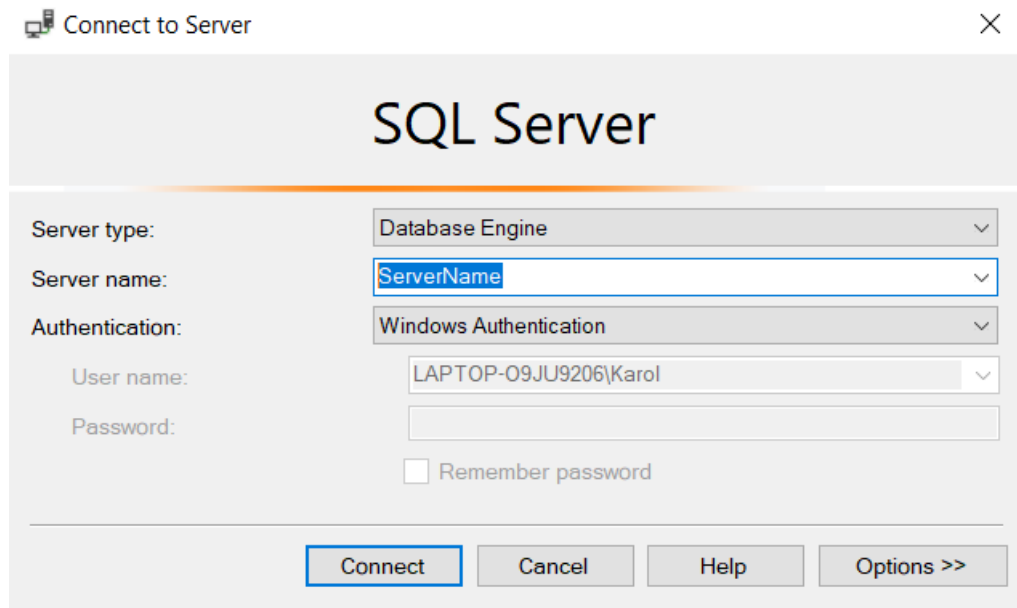
- Wykres



5. Implementacja i testy aplikacji

5.1. Instalacja i konfigurowanie systemu

- Wymagany system operacyjny : Windows 10.
- Pobierz i zainstaluj SQL Server oraz SQL Server Managment Studio.
- Połącz się z serwerem oraz zapisz jego nazwę gdyż będzie później potrzebna.





- Uruchom skrypt tworzący bazę danych .

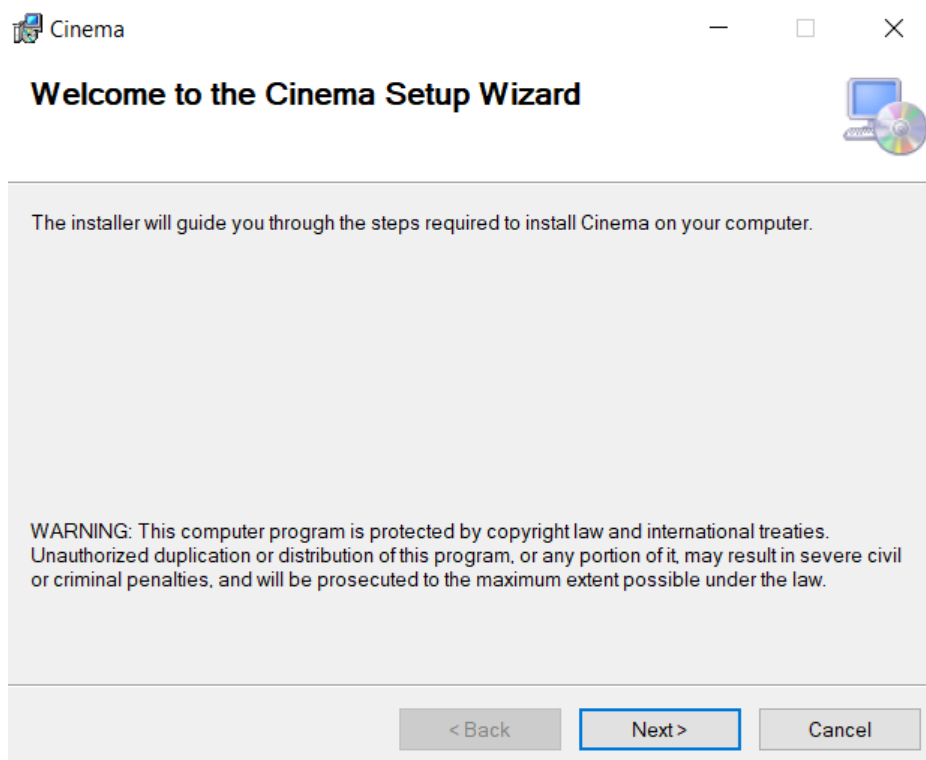
```
USE [master]
GO
/***** Object: Database
CREATE DATABASE [Cinema]
```



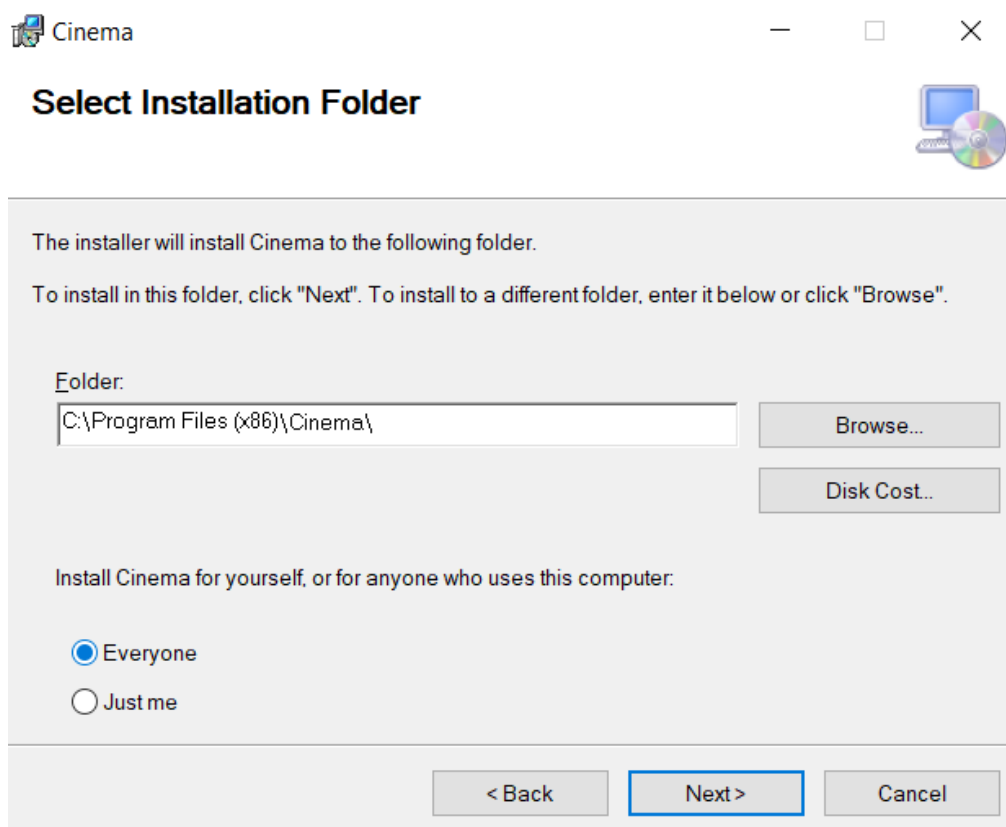
- Uruchom aplikację setup w celu zainstalowania aplikacji.

Nazwa	Data modyfikacji	Typ	Rozmiar
 CinemaSetup	14.01.2020 19:05	Pakiet Instalatora ...	3 268 KB
 setup	14.01.2020 19:05	Aplikacja	772 KB

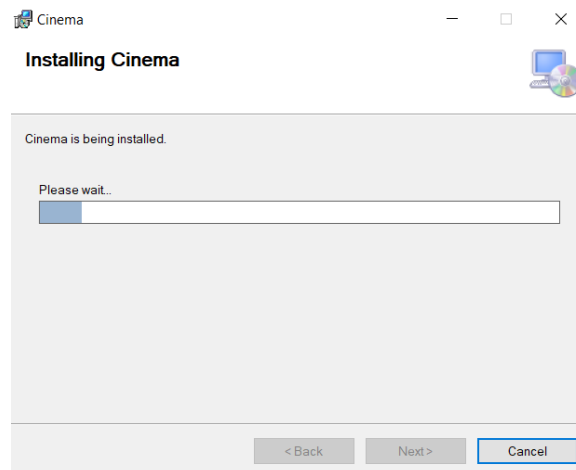
- Na ekranie powinien pojawić się instalator :



- Następnie wybierz folder docelowy :



- Poczekaj aż proces instalacji dobiegnie końca



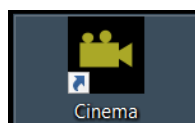
- Przejdź do folderu, który wybrałeś do instalacji i edytuj plik Cinema.exe.config wstawiając nazwę swojego serwera w datasource tak jak pokazano na rysunku :

```

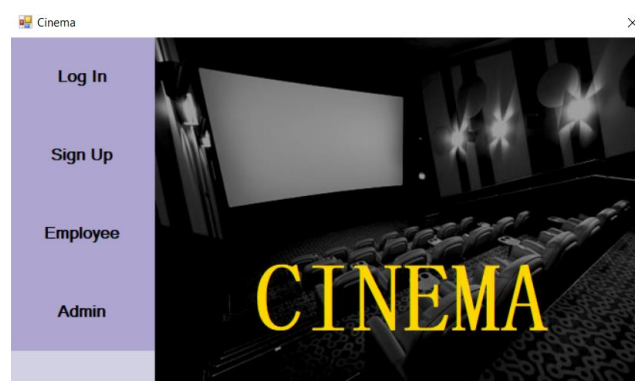
1 <?xml version="1.0" encoding="utf-8"?>
2 <configuration>
3   <configSections>
4     <!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?linkID=237468 -->
5     <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0, Culture=neutral,
6       PublicKeyToken=b77a5c561934e089" requirePermission="false" />
7   </configSections>
8   <startup>
9     <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
10  </startup>
11  <connectionStrings>
12    <add name="CinemaEntities" connectionString="metadata=res:///Database.CinemaModel.csdl;res:///Database.CinemaModel.ssdl;res:///Database.CinemaModel.msl;provider=System.Data.SqlClient;provider connection string="data
13    source=ServerName;initial catalog=Cinema;integrated security=True;MultipleActiveResultSets=True;App=EntityFramework"; providerName="System.Data.EntityClient" />
14  </connectionStrings>
15  <entityFramework>
16    <defaultConnectionFactory type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory, EntityFramework">
17      <parameters>
18        <parameter value="mssqllocaldb" />
19      </parameters>
20    </defaultConnectionFactory>
21    <providers>
22      <provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />
23    </providers>
24  </entityFramework>
25 </configuration>

```

- Następnie kliknij na ikonę aplikacji.

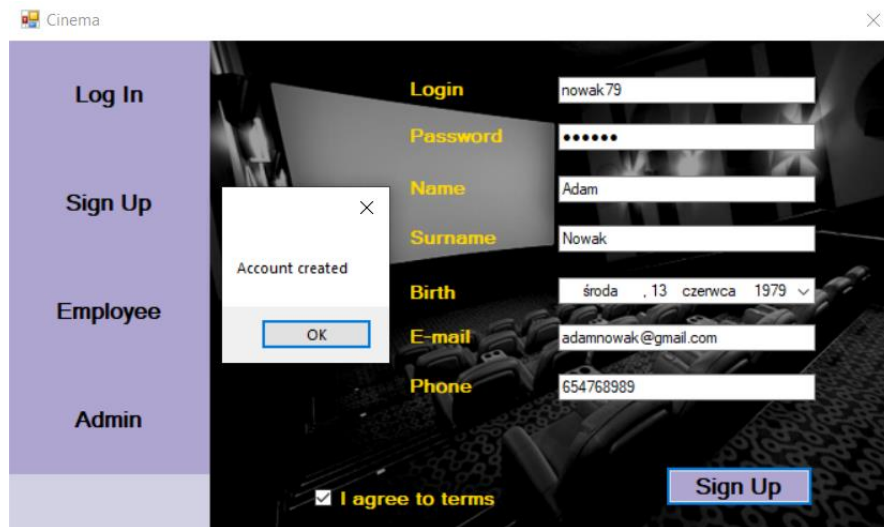


- Na ekranie powinien ukazać się ekran startowy.



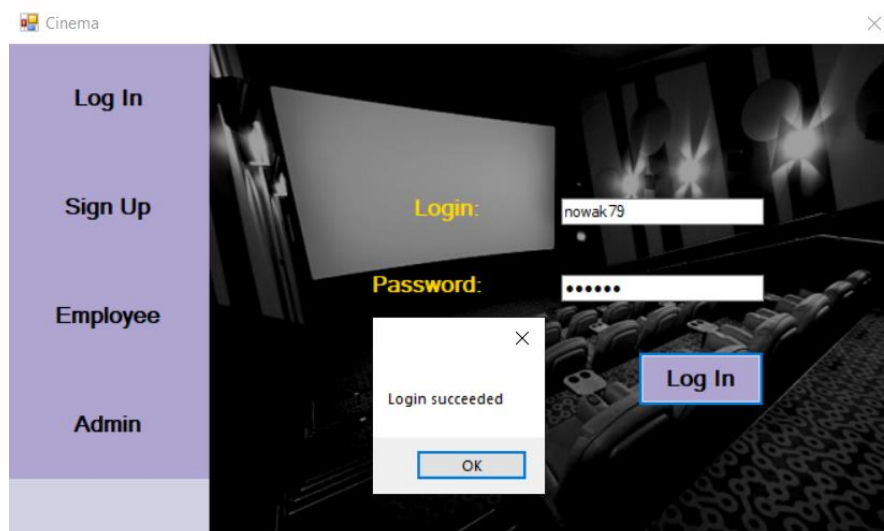
5.2. Instrukcja użytkowania aplikacji oraz testowanie funkcji systemu

- Żeby korzystać z aplikacji wymagane jest założenia konta w systemie.



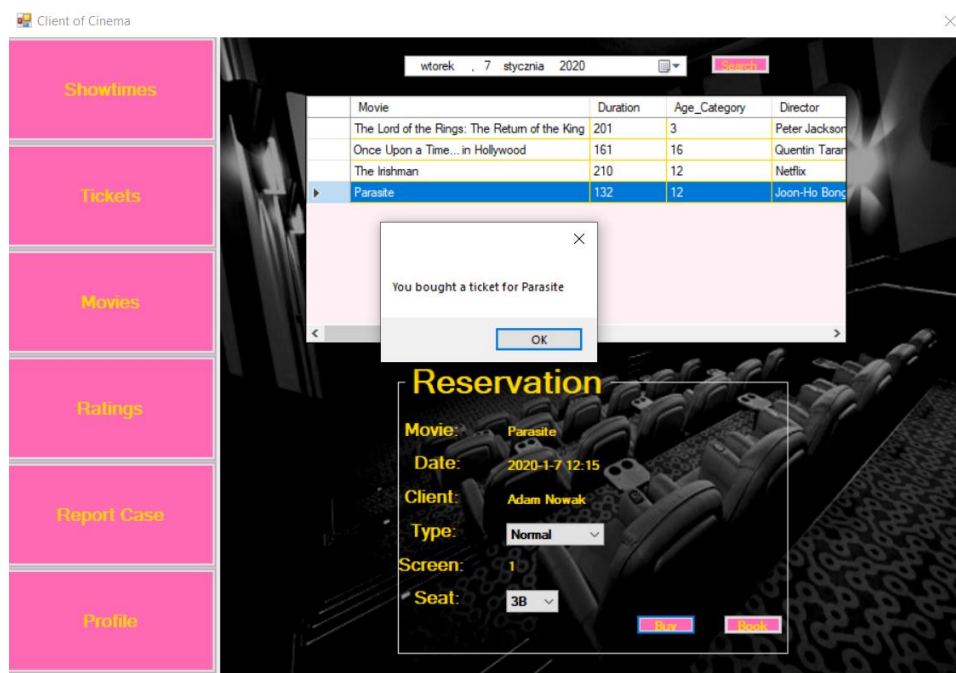
The screenshot shows the Cinema application interface. On the left is a purple sidebar with buttons: Log In, Sign Up, Employee, and Admin. The main area has a dark background with a cinema interior. It contains a 'Sign Up' form with fields for Login (nowak79), Password (masked with dots), Name (Adam), Surname (Nowak), Birth (dropdown showing 'środa, 13 czerwca 1979'), E-mail (adamnowak@gmail.com), and Phone (654768989). There is a 'Sign Up' button at the bottom right and a checkbox for 'I agree to terms'. A white dialog box in the center says 'Account created' with an 'OK' button.

- Następnie należy się zalogować podając login oraz hasło.

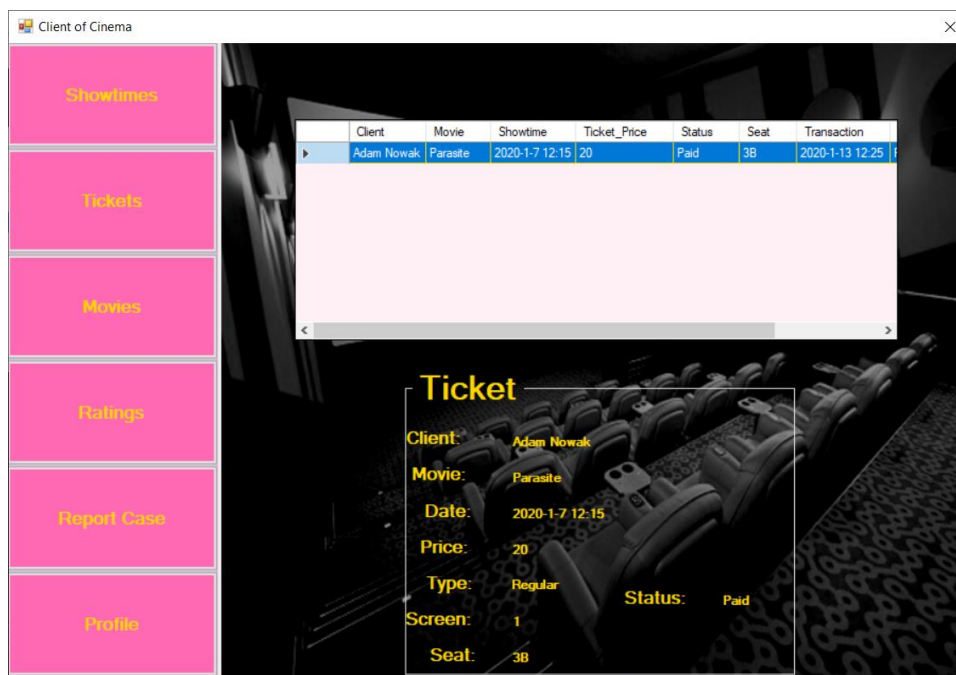


The screenshot shows the Cinema application interface. On the left is a purple sidebar with buttons: Log In, Sign Up, Employee, and Admin. The main area has a dark background with a cinema interior. It contains a 'Login' form with fields for Login (nowak79) and Password (masked with dots). There is a 'Log In' button at the bottom right. A white dialog box in the center says 'Login succeeded' with an 'OK' button.

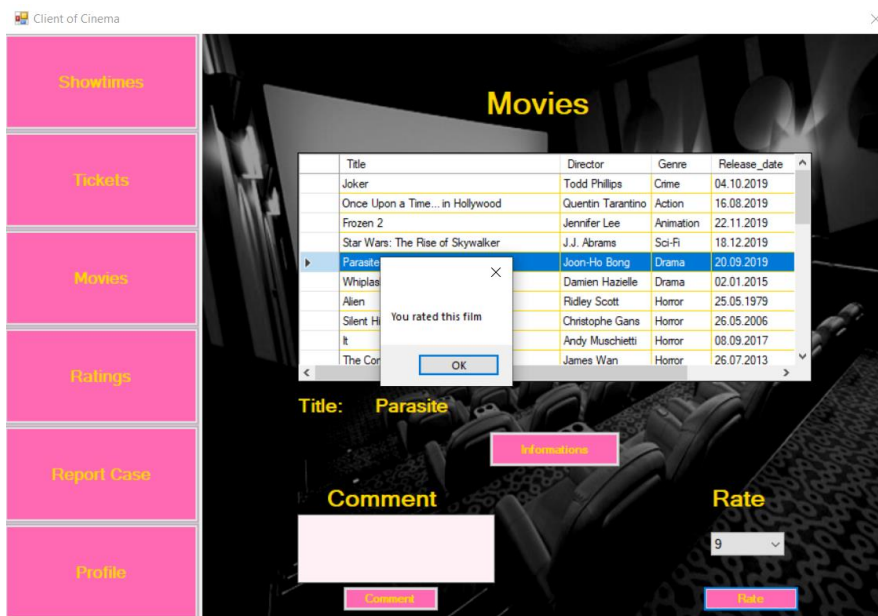
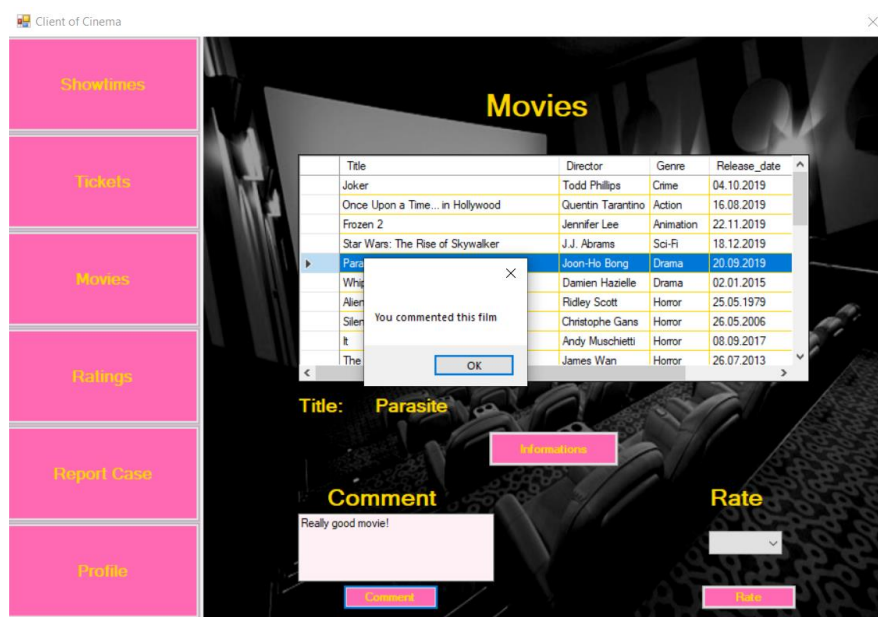
- W zakładce Showtimes po kliknięciu na wybrany z tabeli seans można zakupić lub zarezerwować bilet.



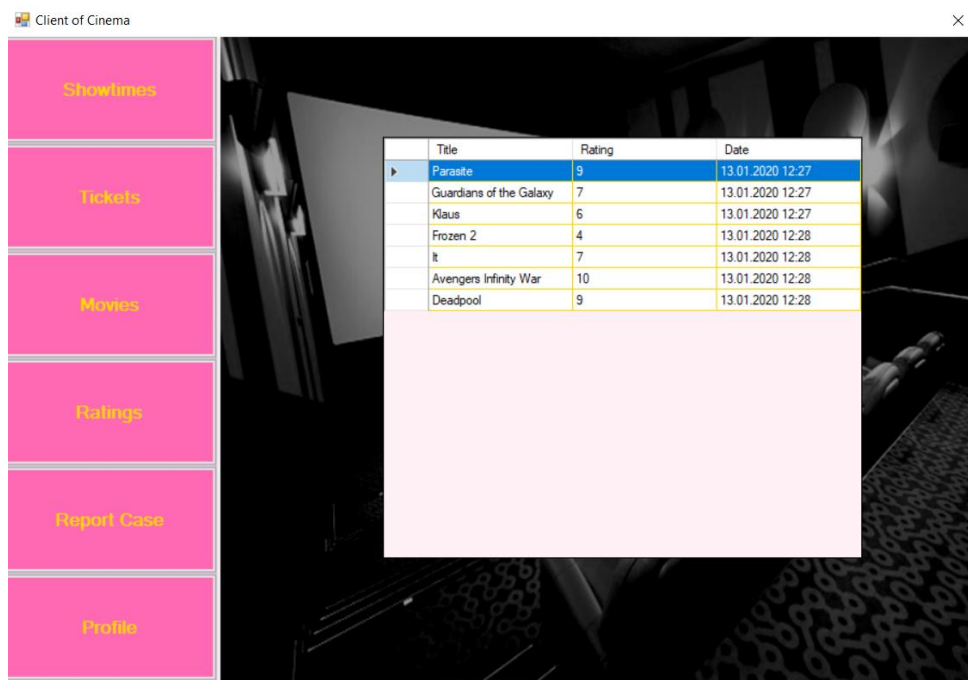
- Posiadane bilety można przeglądać w zakładce Tickets



- W zakładce Movies możemy komentować i oceniać filmy.



- W Ratings możemy zobaczyć nasze oceny

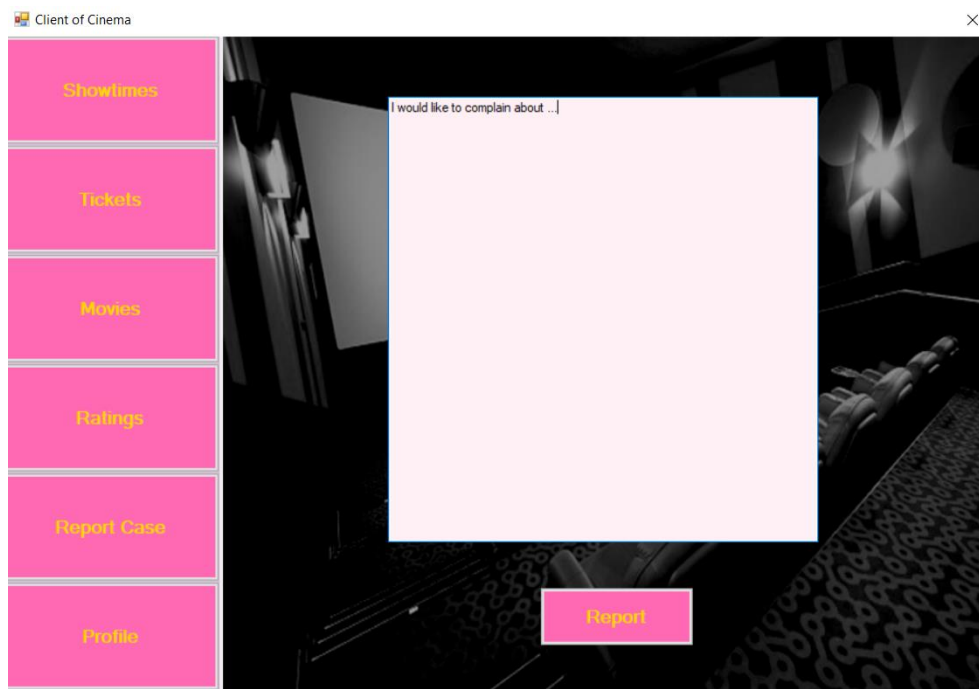


Client of Cinema

Navigation menu: Showtimes, Tickets, Movies, Ratings, Report Case, Profile

Title	Rating	Date
Parasite	9	13.01.2020 12:27
Guardians of the Galaxy	7	13.01.2020 12:27
Klaus	6	13.01.2020 12:27
Frozen 2	4	13.01.2020 12:28
It	7	13.01.2020 12:28
Avengers Infinity War	10	13.01.2020 12:28
Deadpool	9	13.01.2020 12:28

- W celu zgłoszenie skargi należy przejść do zakładki Report Case



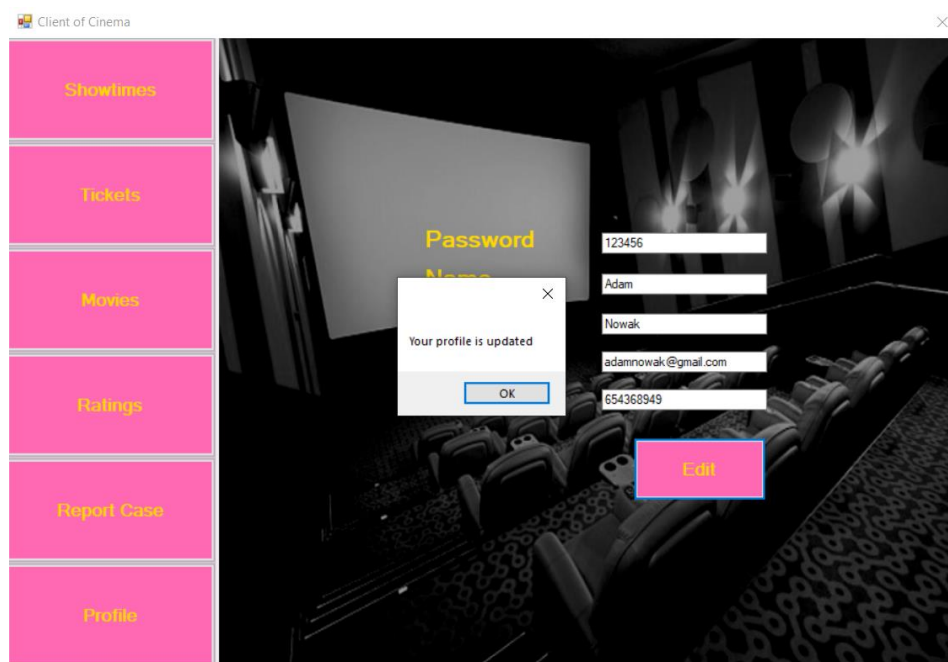
Client of Cinema

Navigation menu: Showtimes, Tickets, Movies, Ratings, Report Case, Profile

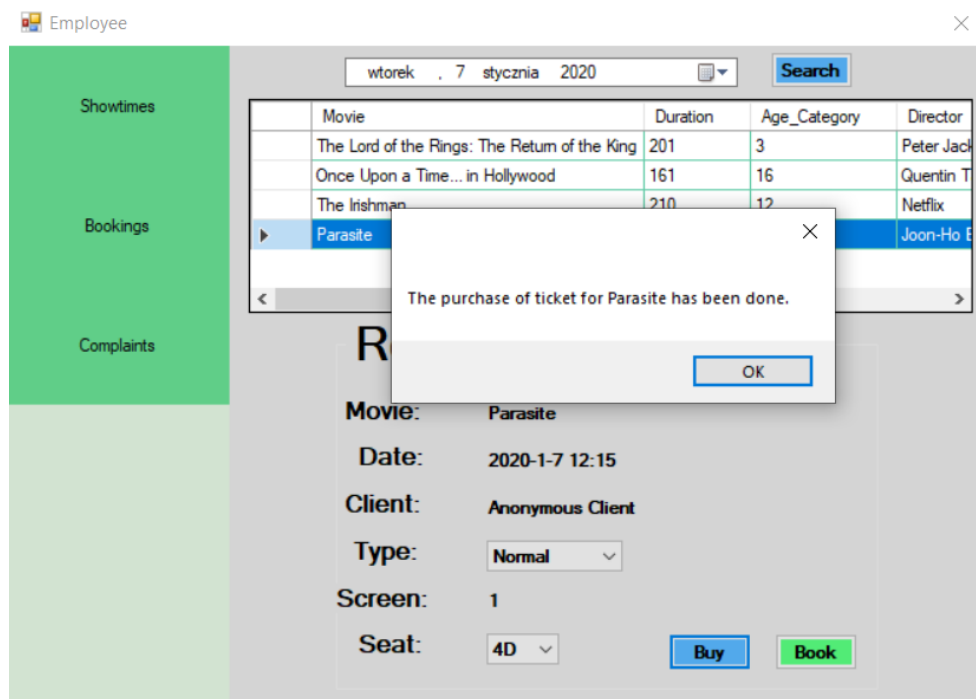
I would like to complain about ...

Report

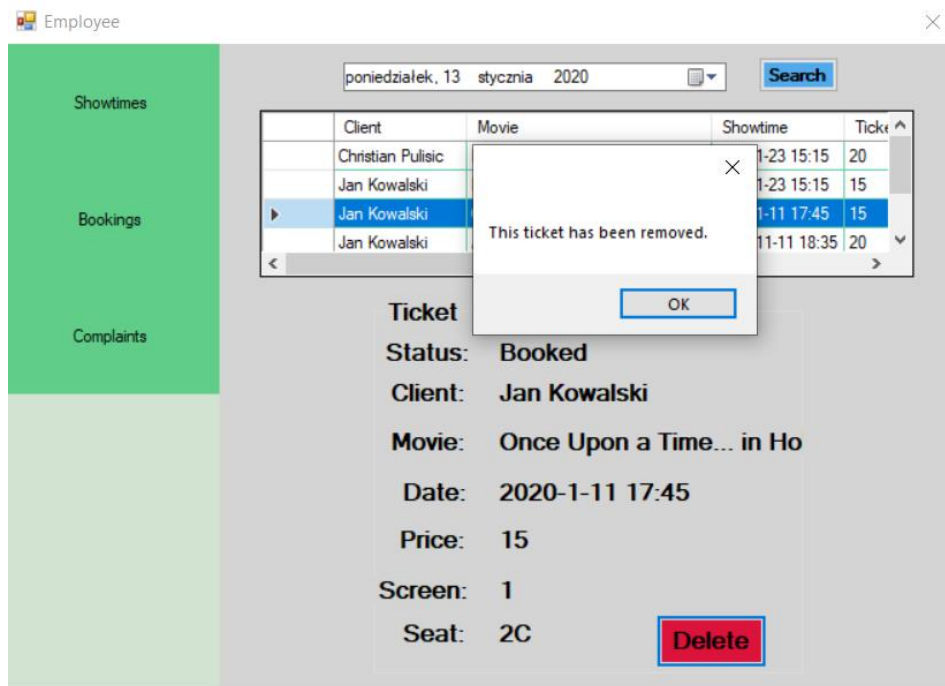
- Edycja profilu możliwa jest w zakładce Profile



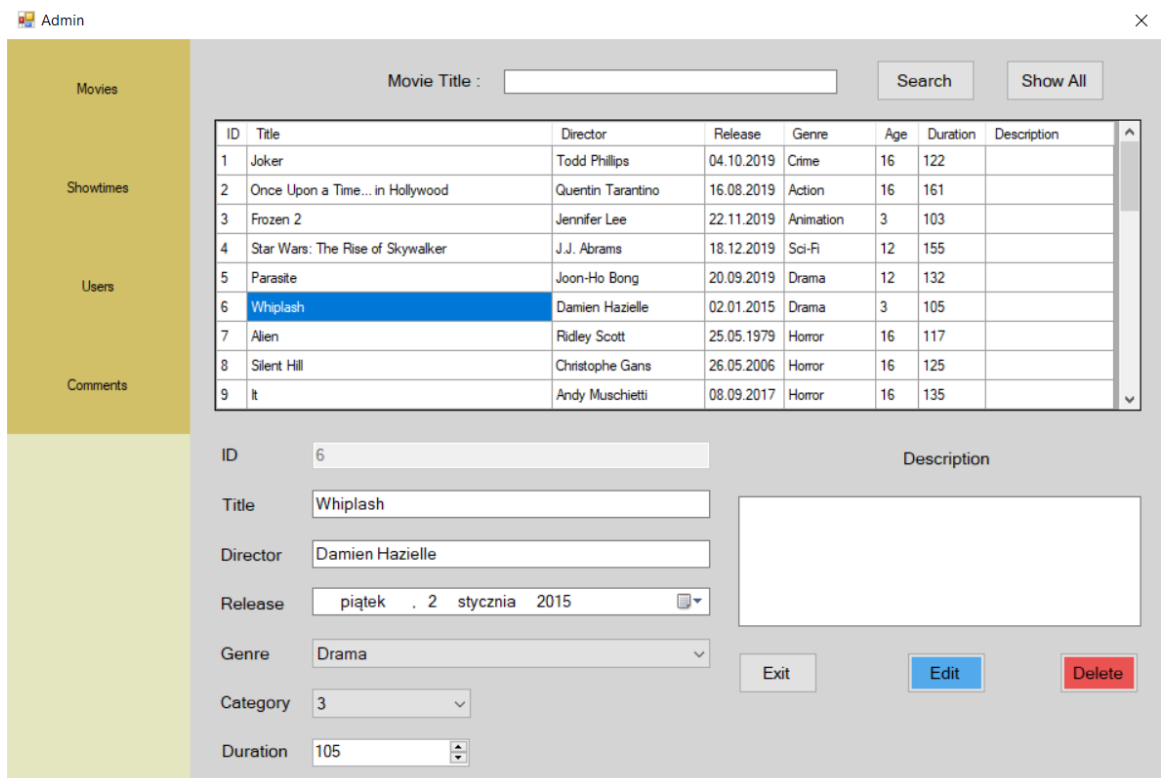
- Pracownik w zakładce Showtimes może kupować i rezerwować bilety na seans w imieniu anonimowego klienta



- Pracownik w celu usunięcia rezerwacji musi przejść do zakładki Bookings



- W przypadku administratora w zakładce Movies może on edytować, dodawać oraz usuwać filmy



- Administrator w celu edycji seansu lub utworzenia nowego musi przejść do zakładki Showtimes

Admin

Movies Showtimes Users Comments

Showtimes Movies

Showtime_ID	Movie_ID	Title	Screen_ID	Date	Technology
1	1	Joker	3	26.11.2019 20:15	2D
2	4	Star Wars: The Rise of Skywalker	1	14.01.2020 15:15	3D
3	2	Once Upon a Time... in Hollywood	2	11.01.2020 17:45	2D
4	3	Frozen 2	2	23.01.2020 15:15	2D
20	2	Once Upon a Time... in Hollywood	1	26.11.2019 20:15	2D
21	2	Once Upon a Time... in Hollywood	3	27.11.2019 20:15	2D
22	4	Star Wars: The Rise of Skywalker	1	11.01.2020 18:15	2D
23	15	Avengers	1	11.11.2020 18:35	3D
24	18	Avengers: End Game	1	29.12.2019 18:35	3D

Edit

Showtime_ID: 22

Movie_ID: 4

Date: 02/13/2020 18:15:00

Screen: 1

Technology: 3D

Edit

Showtime updated

OK

- W zakładce Users administrator może usuwać użytkowników z systemu

Admin

Movies Showtimes Users Comments

User Login: nowak79 Search

User

ID: 110082

Login: nowak79

FirstName: Adam

LastName: Nowak

Birth: 13.06.1979 00:00:00

Phone: 654368949

Email: adamnowak@gmail.com

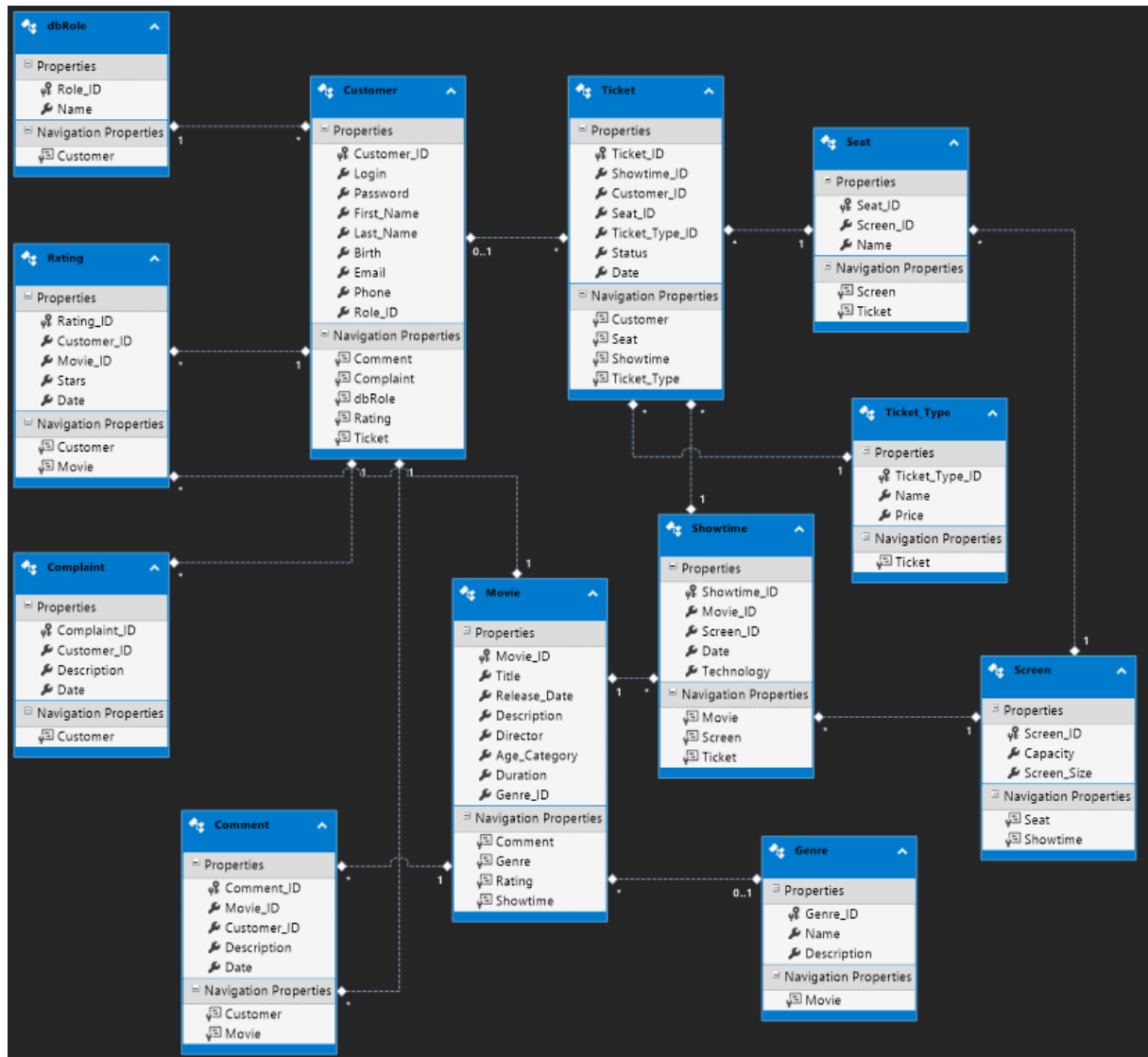
RoleID: 1 Delete

Be careful! All User's tickets, ratings, comments and complaints will be deleted!

5.3. Omówienie wybranych rozwiązań programistycznych

5.3.1. Implementacja interfejsu dostępu do bazy danych

- Dostęp do bazy odbywa się poprzez ORM Entity Framework i technologie LINQ
- Model bazy został zmapowany przez EF do aplikacji



- Zostały utworzone przez EF klasy odpowiadające wszystkim encjom. Przykład klasy Showtime :

```
namespace Cinema.Database
{
    using System;
    using System.Collections.Generic;

    public partial class Showtime
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]
        public Showtime()
        {
            this.Ticket = new HashSet<Ticket>();
        }

        public int Showtime_ID { get; set; }
        public int Movie_ID { get; set; }
        public int Screen_ID { get; set; }
        public System.DateTime Date { get; set; }
        public string Technology { get; set; }

        public virtual Movie Movie { get; set; }
        public virtual Screen Screen { get; set; }
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]
        public virtual ICollection<Ticket> Ticket { get; set; }
    }
}
```

- Funkcjonalności obsługiwane są za pomocą technologii LINQ

5.3.2. Implementacja wybranych funkcjonalności systemu

- Dodanie filmu do bazy (MovieService)

```
public void AddMovie(string title, string director, string release, int genreID, int age, int duration, string description)
{
    using (CinemaEntities database = new CinemaEntities())
    {
        Movie newMovie = database.Movie.Create();

        newMovie.Movie_ID = Convert.ToInt32(database.sp_getSeqMovieID().FirstOrDefault());
        newMovie.Title = title;
        newMovie.Director = director;
        newMovie.Age_Category = age;
        newMovie.Duration = duration;
        newMovie.Release_Date = DateTime.Parse(release);
        newMovie.Genre_ID = genreID;
        newMovie.Description = description;

        database.Movie.Add(newMovie);
        database.SaveChanges();
    }
}
```

- Usuwanie komentarzy (CommentService)

```
public void DeleteComment(int commentID)
{
    using (CinemaEntities database = new CinemaEntities())
    {
        Comment comment = database.Comment.Single(x => x.Comment_ID == commentID);
        database.Comment.Remove(comment);
        database.SaveChanges();
    }
}
```

- Rejestracja użytkownika (CustomerService)

```

LogInAdminPanel.cs [Design]  CustomerService.cs  TicketService.cs  ShowtimesService.cs  MoviesService.cs
Cinema.Services.CustomerService  SignUp(string login, string password, string firstName, str

}

public void SignUp(string login, string password, string firstName, string lastName, string birth, string email, string phone)
{
    using (CinemaEntities database = new CinemaEntities())
    {
        Customer newCustomer = database.Customer.Create();

        newCustomer.Customer_ID = Convert.ToInt32(database.sp_getSeqCustomerID().FirstOrDefault());
        newCustomer.Login = login;
        newCustomer.Password = password;
        newCustomer.First_Name = firstName;
        newCustomer.Last_Name = lastName;
        newCustomer.Birth = DateTime.Parse(birth);
        newCustomer.Email = email;
        newCustomer.Phone = phone;
        newCustomer.Role_ID = 1;

        // jeśli zajęty
        if (CustomerExist(login))
        {
            MessageBox.Show("User name is already taken");
        }
        // jeśli nie jest zajęty
        else
        {
            database.Customer.Add(newCustomer);
            database.SaveChanges();
            MessageBox.Show("Account created");
        }
    }
}

```

- Zakup/Rezerwacja biletu (TicketService)

```

public void OrderingTicket( int showtimeID, int customerID, int seatID, int ticketTypeID, string status)
{
    using (CinemaEntities database = new CinemaEntities())
    {
        Ticket newTicket = database.Ticket.Create();

        //obecna data
        DateTime myDateTime = DateTime.Now;
        //format daty poprawny do sql
        string sqlFormattedDate = myDateTime.ToString("yyyy-MM-dd HH:mm");

        newTicket.Ticket_ID = Convert.ToInt32(database.sp_getSeqTicketID().FirstOrDefault());
        newTicket.Customer_ID = customerID;
        newTicket.Showtime_ID = showtimeID;
        newTicket.Seat_ID = seatID;
        newTicket.Ticket_Type_ID = ticketTypeID;
        newTicket.Status = status;
        newTicket.Date = DateTime.Parse(sqlFormattedDate);

        database.Ticket.Add(newTicket);
        database.SaveChanges();
    }
}

```


5.3.3. Implementacja mechanizmów bezpieczeństwa

- Logowanie do systemu

```
private void buttonLogInAccept_Click(object sender, EventArgs e)
{
    CustomerService service = new CustomerService();
    if (service.LogIn(textBoxLogin.Text, textBoxPassword.Text, 1))
    {
        MessageBox.Show("Login succeeded");
        formMain.Hide();
        customerID= service.GetCustomerID(textBoxLogin.Text);
        //obiekt nowego okna i przekazanie skupienia na drugie okno zamkniecie obecnego
        FormClient formClient = new FormClient(customerID);
        formClient.FormClosed += (s, args) => formMain.Close();
        formClient.Show();
    }
    else
    {
        //okno informujace o blednych danych
        MessageBox.Show("Invalid data provided");
        textBoxLogin.Text = "";
        textBoxPassword.Text = "";
    }
}
```

```
class CustomerService
{
    public bool LogIn(string login, string password, int roleID)
    {
        using (CinemaEntities database = new CinemaEntities())
        {
            bool access = false;

            var signIn = database.Customer.Where(x => x.Login == login
                && x.Password == password && x.Role_ID == roleID).Count();

            if (signIn == 1)
            {
                access = true;
            }

            return access;
        }
    }
}
```

6. Podsumowanie i wnioski

Projekt udało się zrealizować zgodnie z założeniami. Działająca baza danych powstała w systemie SQL Server. Za pomocą środowiska SQL Server Management Studio została ona następnie wykonana w języku SQL (T-SQL) i przetestowana. Następnie wykonaliśmy aplikację desktopową w języku C# z wykorzystaniem Entity Framework jako ORM. Aplikacja została także przetestowana i działa poprawnie.

System SQL Server oraz środowisko SQL Server Management Studio są proste i przyjazne w obsłudze oraz pozwalają na wygodne tworzenie baz danych przy użyciu języka T-SQL czyli rozszerzenia języka SQL.

Technologia Windows Forms oraz język C# pozwala na proste zbudowanie aplikacji desktopowej z intuicyjnym graficznym interfejsem, do jego obsługi przeznaczonej na system operacyjny Windows. W związku z faktem, że język C# jest zorientowany obiektowo warto było skorzystać z narzędzia ORM takiego jak na przykład Entity Framework oraz technologii LINQ służącej do zadawania zapytań na obiektach o składni przypominającej język SQL, gdyż to znacznie ułatwiło obsługi relacyjnej bazy danych ze strony aplikacji.