Cinema Tickets

Adam Czerwiński Wydział Matematyki Stosowanej St. II Sem. II Informatyka

Listopad 2021

1 Opis Problemu

Należy zaprojektować oraz zaimplementować system rezerwacji biletów do kina przedstawionego za pomocą systemu rozproszonego z wykorzystaniem relacyjnej bazy danych. System powinien być możliwy do uruchomienia na kilku instancjach równocześnie.

System powinien mieć następującą funkcjonalność:

- dodawanie nowych filmów oraz ich modyfikacja,
- dodawanie nowych sal oraz ich modyfikacja,
- dodawanie nowych seansów oraz ich modyfikacja,
- dodawanie nowych kont pracowników oraz ich modyfikacja,
- dodawanie nowych kont klientów,
- rezerwowanie miejsc na salach przez klientów,
- zwalnianie miejsc rezerwacji,
- szyfrowanie danych

2 Przedstawienie aplikacji

2.1 Interfejs Logowania

Interfejs logowania dzieli się na:

- ekran logowania klienta,
- ekran logowania pracownika.

Klient ma dodatkowo opcję utworzenia konta. Konto musi mieć unikalny login.

Rysunek 1: Ekran logowania dla klienta

Źródło: Opracowanie własne

Pracownik może mieć utworzone konto tylko i wyłącznie dzięki innemu pracownikowi.

Rysunek 2: Ekran logowania dla pracownika

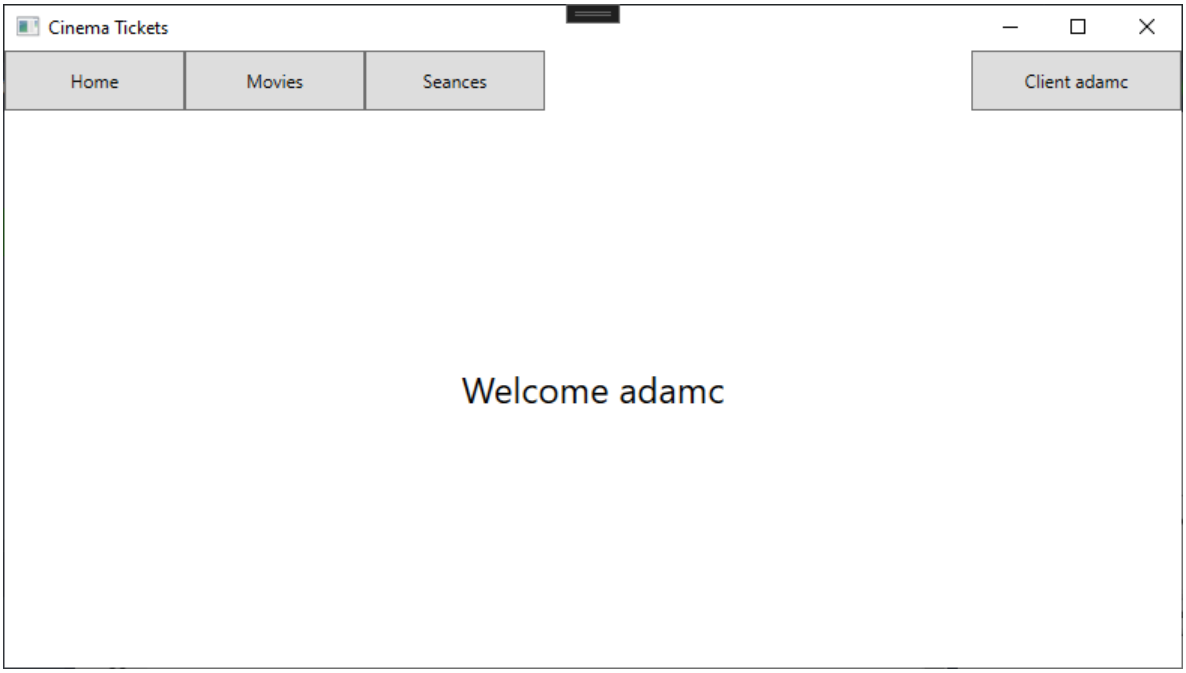
Źródło: Opracowanie własne

Możliwość interakcji użytkownika z przyciskami odpowiadającymi za logowanie oraz tworzenie konta jest ograniczona poprzez pewne wymagania. Między innymi uzupełniony prawidłowy login oraz hasło.

2.2 Główny ekran aplikacji

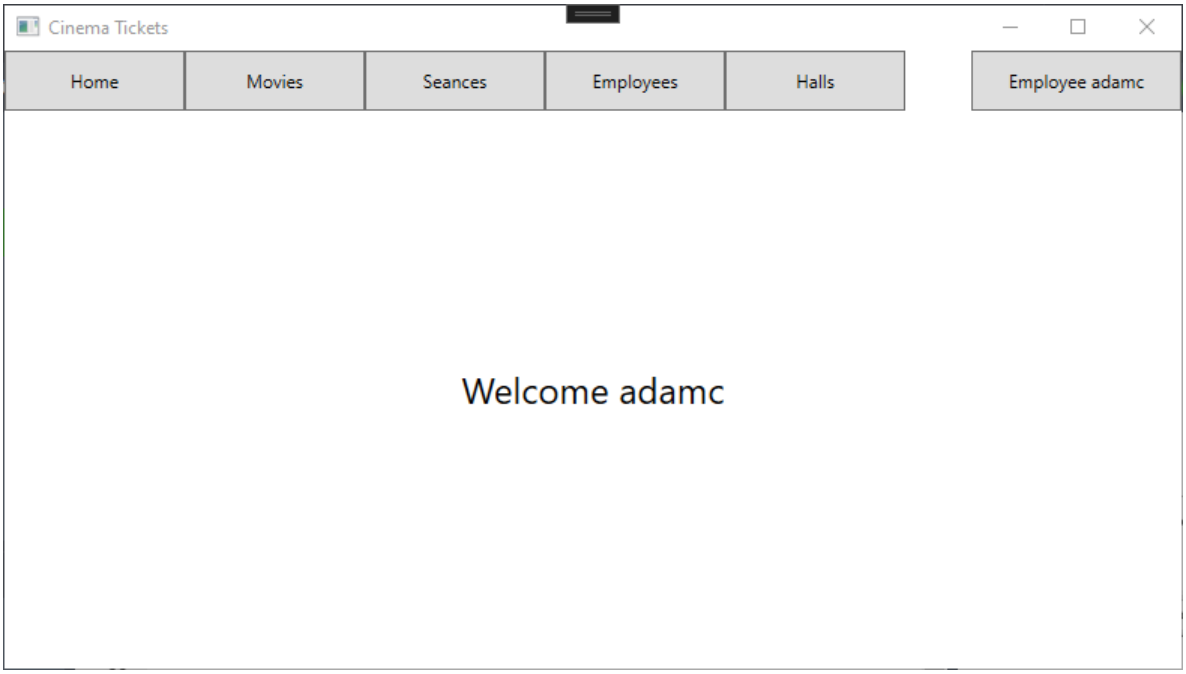
Główny ekran aplikacji widzimy po prawidłowym zalogowaniu przez klienta lub pracownika. Górna część aplikacji składa się z nawigacji użytkownika.

Nawigacja jest dopasowana w zależności od typu konta. Na końcu nawigacji znajduje się przycisk z typem zalogowanego konta oraz jego loginem, który jest odpowiedzialny za wylogowanie z aplikacji. Po wylogowaniu użytkownik zostaje przeniesiony do ekranu logowania.



Rysunek 3: Główny ekran klienta

Źródło: Opracowanie własne



Rysunek 4: Główny ekran pracownika

Źródło: Opracowanie własne

Jak widzimy pracownik ma dodatkowo zakładkę możliwą do zarządzania pracownikami oraz salami kinowymi.

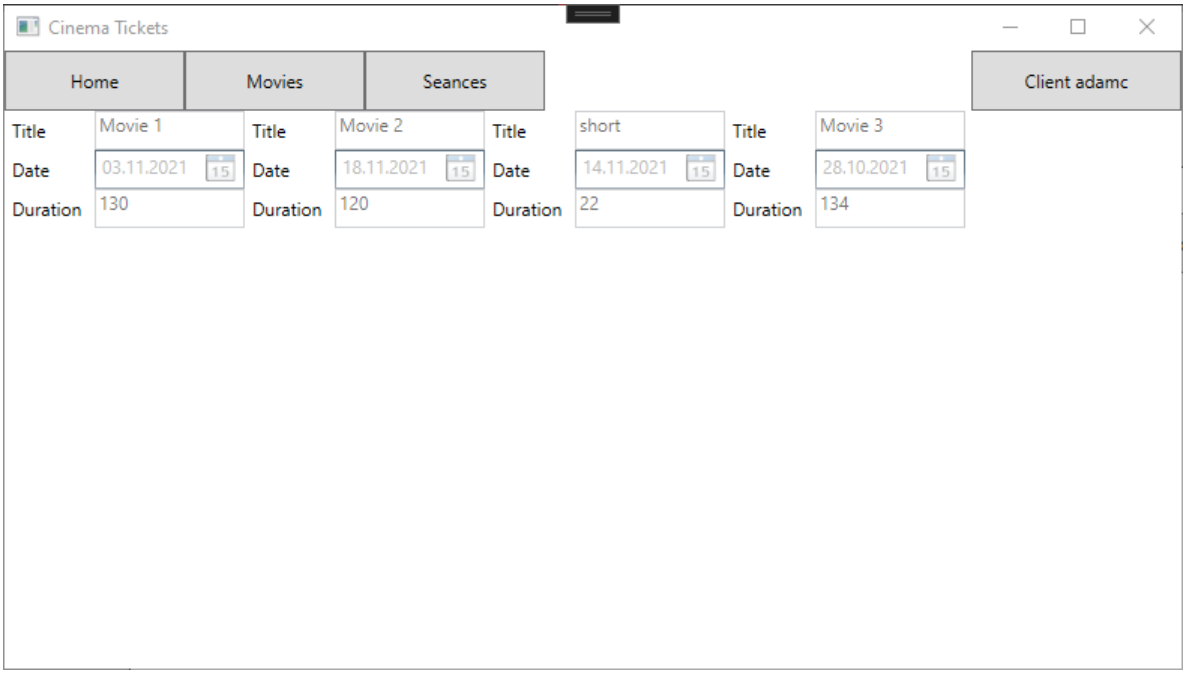
2.3 Filmy

Możliwość zarządzania filmami jest ograniczona tylko dla pracowników. Klienci mogą jedynie przeglądać filmy.

Pracownik może:

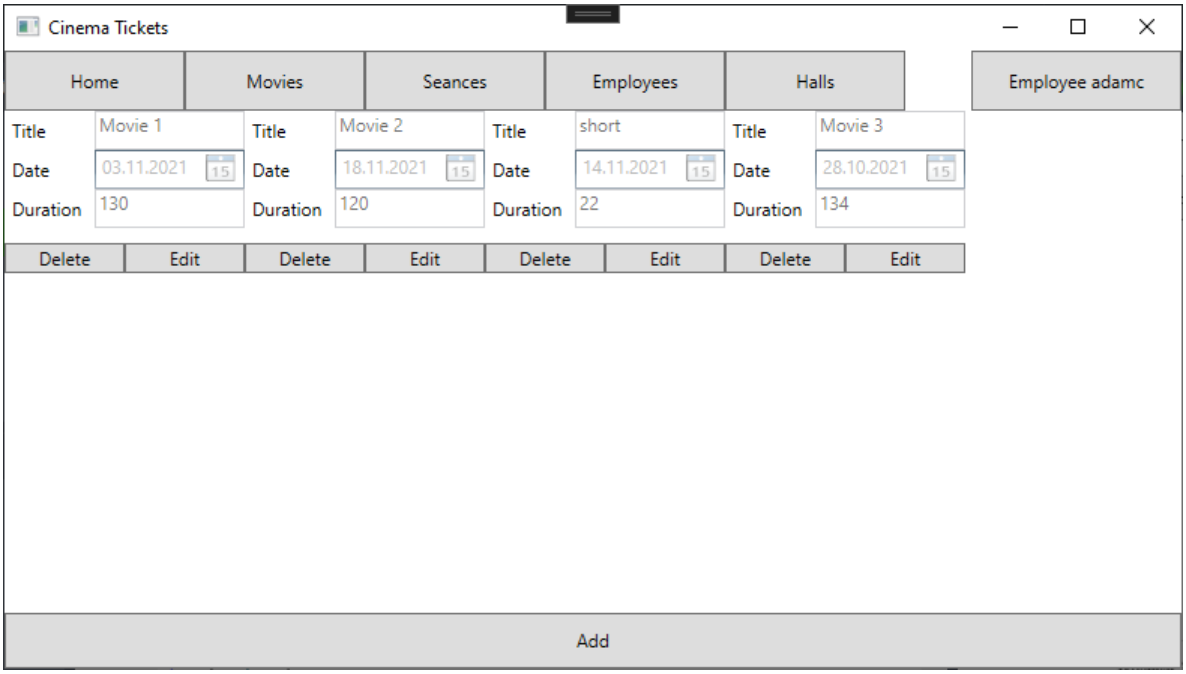
- dodawać filmy,
- modyfikować filmy,
- usuwać filmy.

Filmy można usuwać tylko jeżeli nie są one wykorzystywane (np. trwa seans). Film składa się z tytułu, daty wypuszczenia oraz długości trwania przedstawionego w minutach.



Rysunek 5: Filmy widoczne dla klienta

Źródło: Opracowanie własne



Rysunek 6: Filmy widoczne dla pracownika

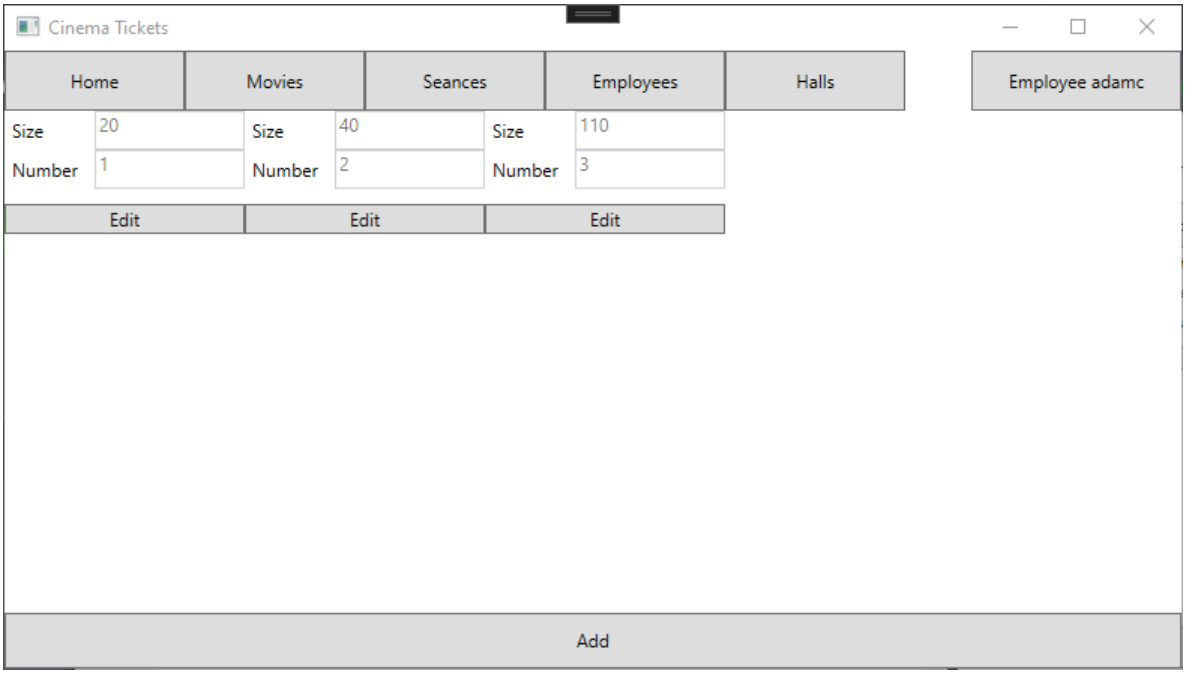
Źródło: Opracowanie własne

2.4 Sale kinowe

Ekran zawierający sale kinowe możliwy jest tylko do zobaczenia przez pracowników. Sale kinowe składają się z numeru sali oraz z ilości miejsc.

Salami kinowymi można zarządzać w następujący sposób:

- dodawanie nowych sal kinowych z unikalnym numerem sali,
- modyfikacja sal kinowych.



Rysunek 7: Widok sal kinowych

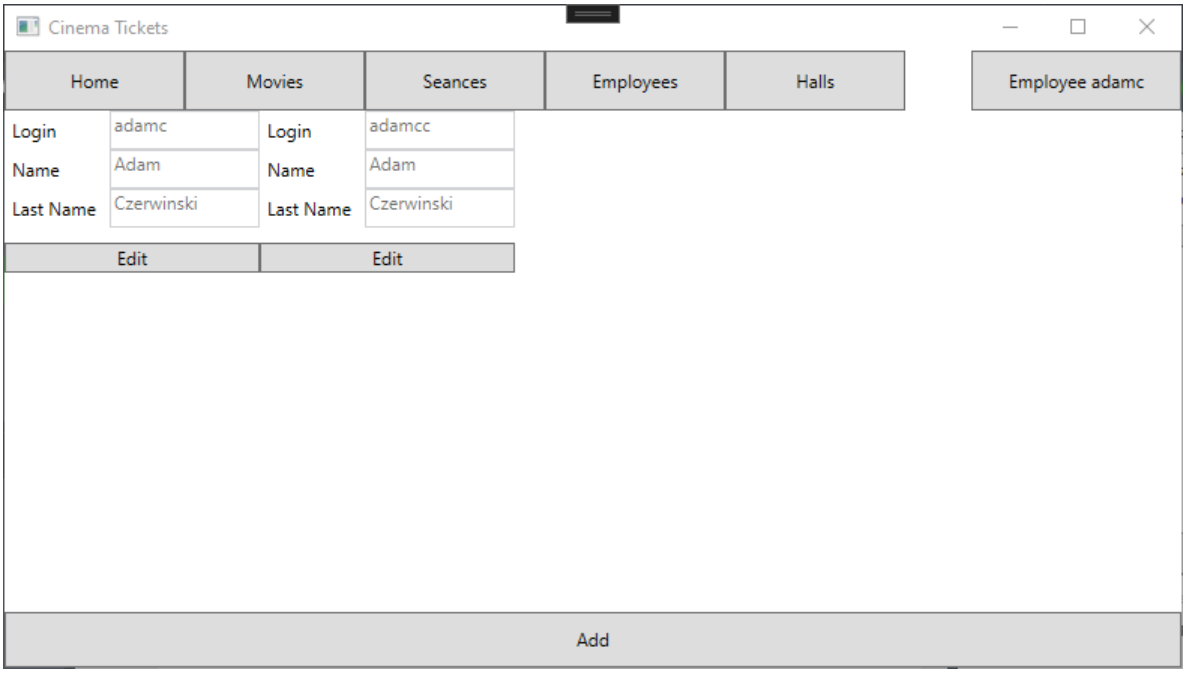
Źródło: Opracowanie własne

2.5 Pracownicy

Pracownikami może zarządzać i przeglądać tylko inny pracownik. Każdy z pracowników ma unikalny login. Zawiera on podstawowe dane jak imię i nazwisko.

Pracownikami można zarządzać w następujący sposób:

- dodawanie nowych pracowników,
- modyfikacja istniejących pracowników (za wyjątkiem loginu).



Rysunek 8: Widok pracowników

Źródło: Opracowanie własne

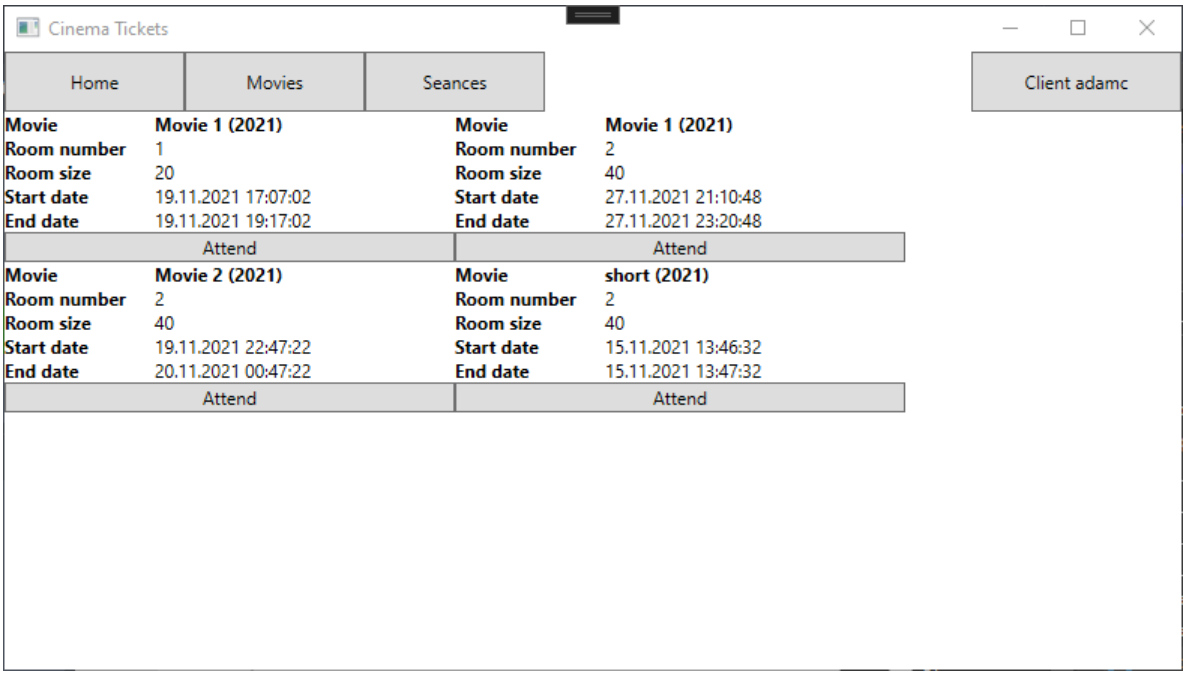
2.6 Seansy filmowe

Seansy filmowe może zobaczyć pracownik jak i klient, jednakże są pewne różnice. Pracownik ma tylko i wyłącznie możliwość dodawania nowych seansów filmowych, a klient tylko możliwość dołączenia lub anulowania dołączenia do nich.

Dla danego filmu i danej sali może istnieć tylko jeden seans.

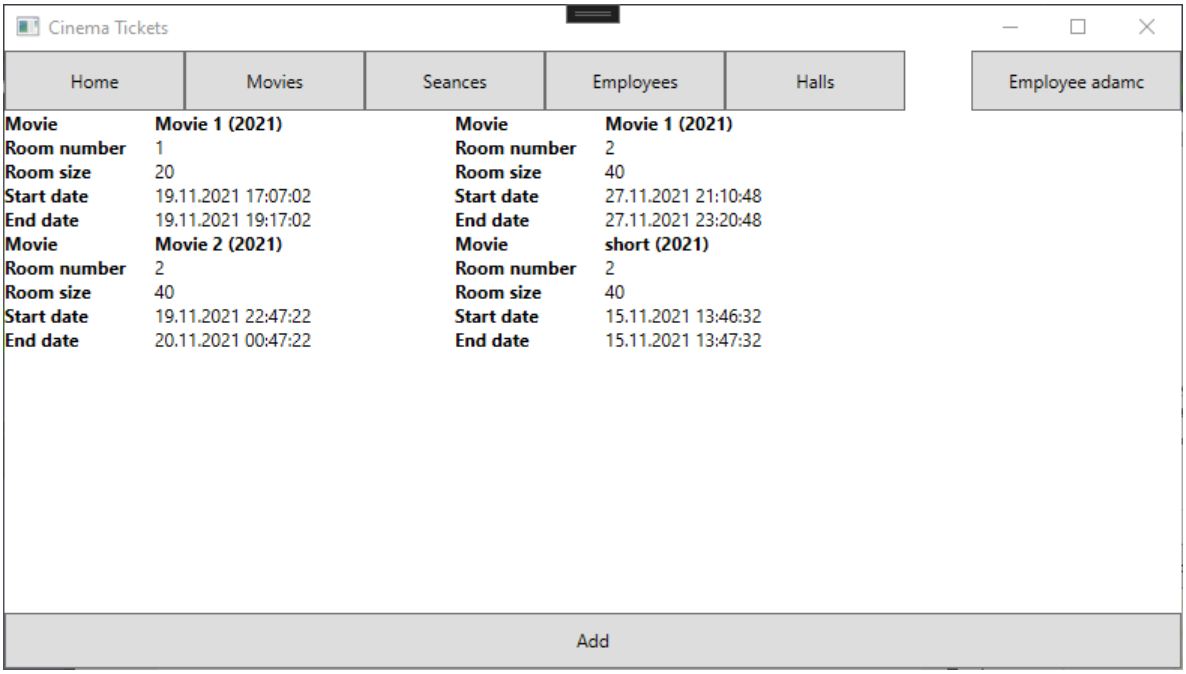
Widoczne są tylko seansy, których data zakończenia jest późniejsza niż aktualna data.

Wszystkie przedawnione (z datą wcześniejszą niż aktualna) seansy są usuwane co określony czas.



Rysunek 9: Seansy widoczne przez klientów

Źródło: Opracowanie własne



Rysunek 10: Seansy widoczne przez pracowników

Źródło: Opracowanie własne

3 Opis systemu

3.1 Wdrożenie

Aplikacja została utworzona za pomocą następujących frameworków:

- Microsoft.NETCore.App w wersji 6.0.0-rc.2.21480.5,
- Microsoft.WindowsDesktop.APP.WPF w wersji 6.0.0-rc.2.21501.6.

Dodatkowo zostały wykorzystane pakiety takie jak:

- Extended.Wpf.Toolkit w wersji 4.1.0,
- Microsoft.EntityFrameworkCore.SqlServer w wersji 5.0.11,
- Microsoft.EntityFrameworkCore.Tools w wersji 5.0.11,
- Microsoft.Extensions.DependencyInjection w wersji 5.0.2,
- System.Reactive w wersji 5.0.0.

Relacyjna baza do przechowywania danych to Microsoft SQL Server 2019 w wersji Express. Podczas tworzenia aplikacji użyto Docker'a oraz obrazu o identyfikatorze *80bdc8efc889*
i nazwie *mcr.microsoft.commssqlserver*.

Do uruchomienia aplikacji wymagany jest składnik **.NET 6.0 Runtime**.

3.2 System rozproszony

Aplikacja jest oparta na systemie rozproszonym, którego zadaniem jest udostępnienie wspólnych zasobów dla wielu instancji. System taki ma następujące zalety:

- Dzielenie zasobów (dane, urządzenia sprzętowe, jak np. drukarki, dyski).
- Przyśpieszenie obliczeń (dzielenie obciążenia).
- Niezawodność (awaria jednego urządzenia nie powinna uniemożliwiać działania systemu, lecz co najwyżej pogorszyć jego wydajność).
- Komunikacja (np. poczta elektroniczna).
- Ekonomiczność (system rozproszony może być tańszy niż odpowiadający mu mocą obliczeniową system scentralizowany)
- Wewnętrzne rozproszenie (niektóre aplikacje są z natury rozproszone i wymagają rozproszonych komputerów).
- Stopniowy wzrost (można stopniowo zwiększać moc obliczeniową systemu; skalowalność to zdolność systemu do adaptowania się do wzrastających zapotrzebowań).

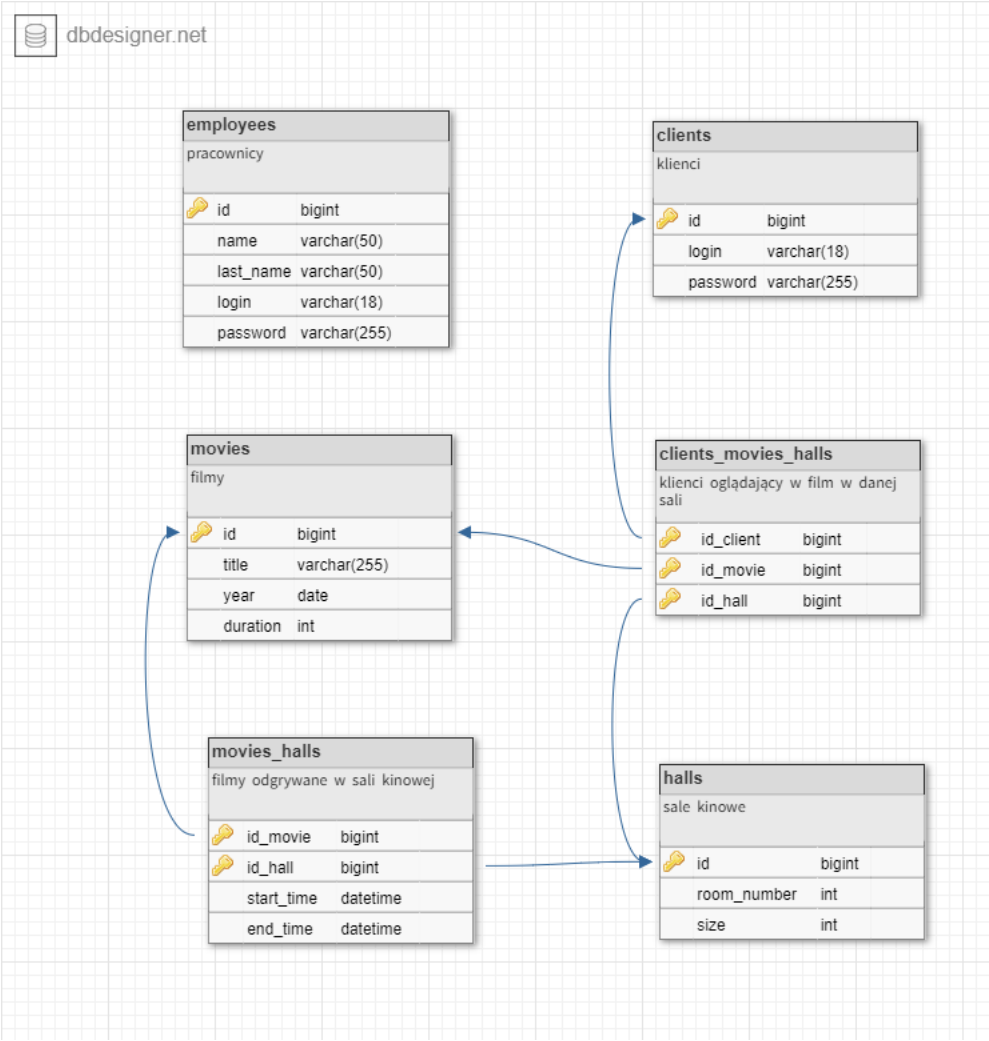
Jak i następujące wady:

- Oprogramowanie (zdecydowanie bardziej złożone; wymaga opracowania wspólnych standardów).
- Sieć (może ulec awarii lub zostać przeciążona).
- Bezpieczeństwo (komputer podłączony do sieci jest mniej bezpieczny).

3.3 Przechowywanie danych

Dane są przechowywanie w relacyjnej bazie Microsoft SQL Server 2019 w wersji Express co umożliwia dostęp do danych z różnych stanowisk komputerowych. Podczas implementacji instancja bazy była przechowywana na środowisku utworzonym za pomocą docker'a w lokalnej części systemu developera.

Model danych składa się z 6 tabel:



Rysunek 11: Model tabel

Źródło: Opracowanie własne

3.3.1 Szyfrowanie danych

Hasło klienta jak i pracownika jest szyfrowane za pomocą **szyfru cezara**. Szyfr cezara polega na przestawieniu każdej litery danego alfabetu o pewną stałą ilość znaków.

Załóżmy, że naszą stałą liczbą przekształcającą słowo jest 3, a dane słowo to *adam*:
adam -> dgdp

Taki szyfr jest łatwy do złamania i nie oferuje on żadnego zabezpieczenia danych. W zamian tego jest szybki i prosty w użyciu.

3.3.2 Tworzenie bazy

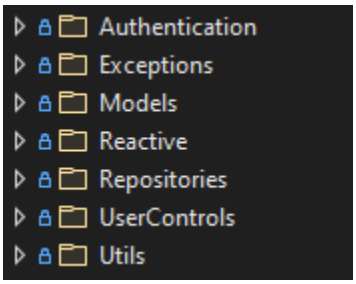
Poniższym skryptem można utworzyć bazę danych wymaganą do poprawnego działania aplikacji:

```
CREATE TABLE [employees] (
id bigint IDENTITY(1,1),
name varchar(50) NOT NULL,
last_name varchar(50) NOT NULL,
login varchar(18) NOT NULL UNIQUE,
password varchar(255) NOT NULL,
  CONSTRAINT [PK_EMPLOYEES] PRIMARY KEY CLUSTERED
  (
  [id] ASC
  ) WITH (IGNORE_DUP_KEY = OFF)

)
GO
CREATE TABLE [clients] (
id bigint IDENTITY(1,1),
login varchar(18) NOT NULL UNIQUE,
password varchar(255) NOT NULL,
  CONSTRAINT [PK_CLIENTS] PRIMARY KEY CLUSTERED
  (
  [id] ASC
  ) WITH (IGNORE_DUP_KEY = OFF)

)
GO
CREATE TABLE [movies] (
id bigint IDENTITY(1,1),
title varchar(255) NOT NULL,
year date NOT NULL,
duration int NOT NULL,
  CONSTRAINT [PK_MOVIES] PRIMARY KEY CLUSTERED
  (
  [id] ASC
  ) WITH (IGNORE_DUP_KEY = OFF)

)
GO
CREATE TABLE [halls] (
id bigint IDENTITY(1,1),
room_number int NOT NULL,
size int NOT NULL,
  CONSTRAINT [PK_HALLS] PRIMARY KEY CLUSTERED
  (
  [id] ASC
  ) WITH (IGNORE_DUP_KEY = OFF)

)
GO
CREATE TABLE [movies_halls] (
id_movie bigint NOT NULL,
id_hall bigint NOT NULL,
start_time datetime NOT NULL,
```

```
end_time datetime NOT NULL,
  CONSTRAINT [PK_MOVIES_HALLS] PRIMARY KEY CLUSTERED
  (
  [id_movie] ASC, [id_hall] ASC
  ) WITH (IGNORE_DUP_KEY = OFF)

)
GO
CREATE TABLE [clients_movies_halls] (
id_client bigint NOT NULL,
id_movie bigint NOT NULL,
id_hall bigint NOT NULL,
  CONSTRAINT [PK_CLIENTS_MOVIES_HALLS] PRIMARY KEY CLUSTERED
  (
  [id_client] ASC, [id_movie] ASC, [id_hall] ASC
  ) WITH (IGNORE_DUP_KEY = OFF)

)
GO


ALTER TABLE [movies_halls] WITH CHECK ADD CONSTRAINT [movies_halls_fk0] FOREIGN KEY ([id_movie]) REFEREN
ON UPDATE CASCADE
GO
ALTER TABLE [movies_halls] CHECK CONSTRAINT [movies_halls_fk0]
GO
ALTER TABLE [movies_halls] WITH CHECK ADD CONSTRAINT [movies_halls_fk1] FOREIGN KEY ([id_hall]) REFERENC
ON UPDATE CASCADE
GO
ALTER TABLE [movies_halls] CHECK CONSTRAINT [movies_halls_fk1]
GO

ALTER TABLE [clients_movies_halls] WITH CHECK ADD CONSTRAINT [clients_movies_halls_fk0] FOREIGN KEY ([id
ON UPDATE CASCADE
GO
ALTER TABLE [clients_movies_halls] CHECK CONSTRAINT [clients_movies_halls_fk0]
GO
ALTER TABLE [clients_movies_halls] WITH CHECK ADD CONSTRAINT [clients_movies_halls_fk1] FOREIGN KEY ([id
ON UPDATE CASCADE
GO
ALTER TABLE [clients_movies_halls] CHECK CONSTRAINT [clients_movies_halls_fk1]
GO
ALTER TABLE [clients_movies_halls] WITH CHECK ADD CONSTRAINT [clients_movies_halls_fk2] FOREIGN KEY ([id
ON UPDATE CASCADE
GO
ALTER TABLE [clients_movies_halls] CHECK CONSTRAINT [clients_movies_halls_fk2]
GO
```

3.4 Implementacja

Projekt podzielony jest na następujące foldery:

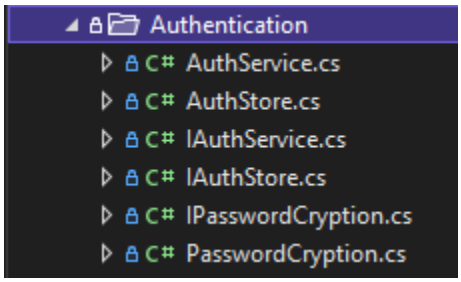


Rysunek 12: Główne drzewo plików

Źródło: Opracowanie własne

3.4.1 Authentication

W tym folderze znajdują się elementy do przechowywania informacji o zalogowanym użytkowniku jak i również umożliwić takiego użytkownika zalogować do systemu.

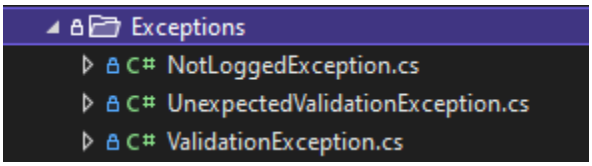


Rysunek 13: Pliki składające się w *Authentication*

Źródło: Opracowanie własne

3.4.2 Exceptions

W tym folderze znajdują się własne wyjątki pomagające obsługiwać pewne przypadki występujące w aplikacji.

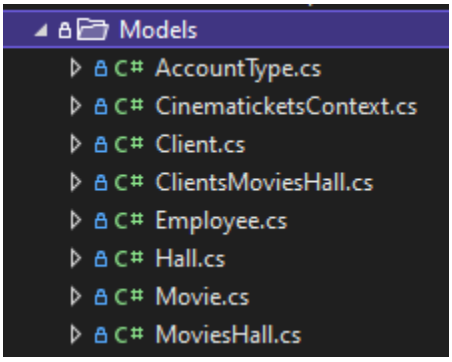


Rysunek 14: Pliki składające się w *Exceptions*

Źródło: Opracowanie własne

3.4.3 Models

W tym folderze znajdują się modele DAO (ang. *Data Object Access*) odzwierciedlające encje w bazie danych.



Rysunek 15: Pliki składające się w *Models*

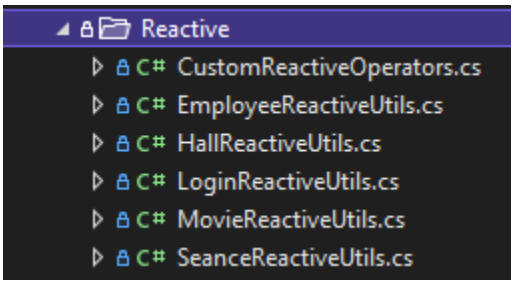
Źródło: Opracowanie własne

Rysunek 16: Diagram UML modelu DAO

Źródło: Opracowanie własne

3.4.4 Reactive

W tym folderze znajdują się pomocnicze klasy odpowiadające za reaktywną część aplikacji i obsługę zdarzeń pomiędzy widokami.



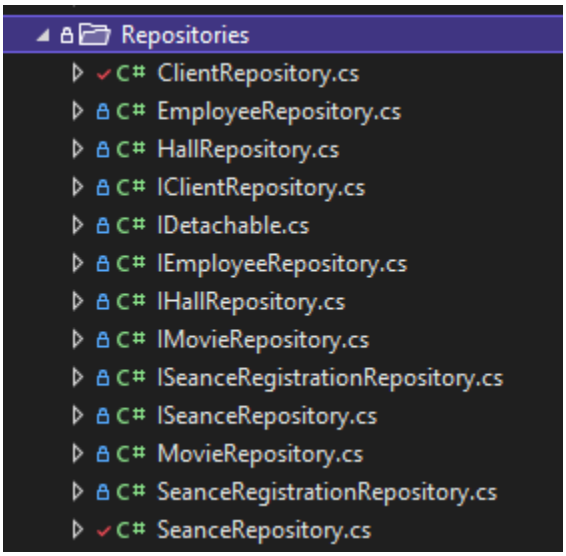
Rysunek 17: Pliki składające się w *Reactive*

Źródło: Opracowanie własne

W części tej wykorzystany został wzorzec obserwatora.

3.4.5 Repositories

W tym folderze znajdują się pomocnicze klasy umożliwiające komunikację ze źródłem naszych danych czyli relacyjną bazą danyc.

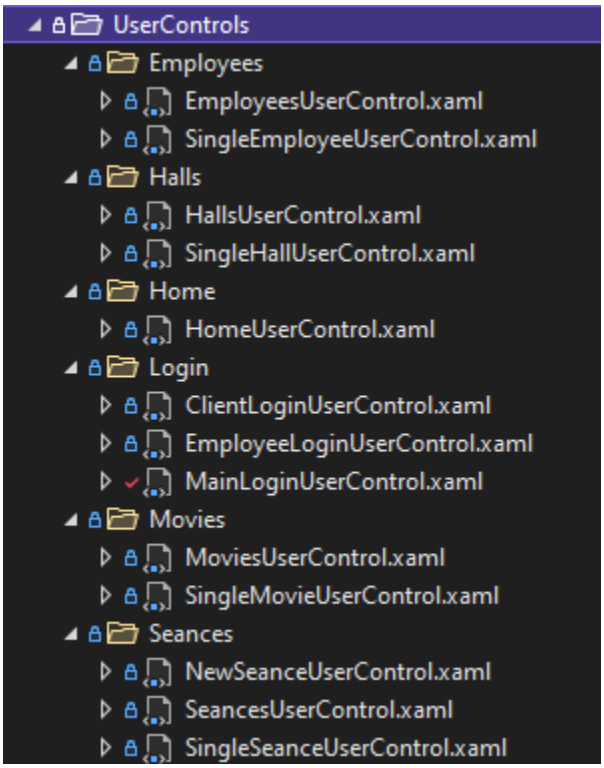


Rysunek 18: Pliki składające się w *Repositories*

Źródło: Opracowanie własne

3.4.6 UserControls

W tym folderze znajdują się widoki ich logika.

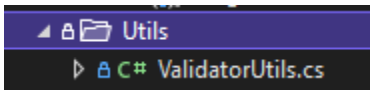


Rysunek 19: Pliki składające się w *UserControls*

Źródło: Opracowanie własne

3.4.7 Utils

W tym folderze znajdują się pomocnicze składniki ogólnego rodzaju,



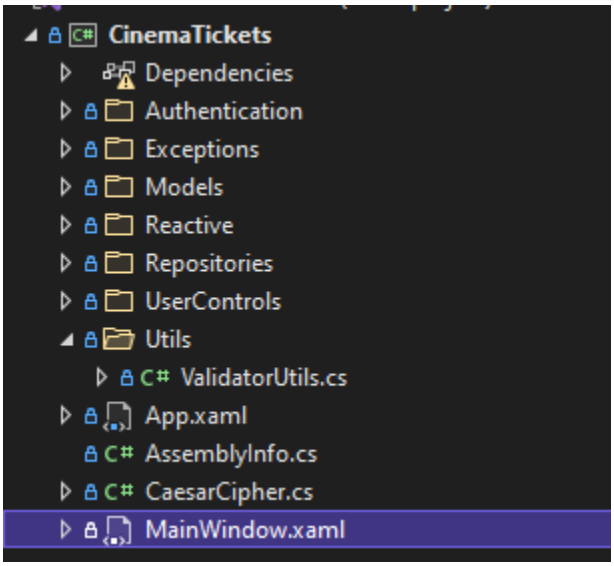
Rysunek 20: Pliki składające się w *Utils*

Źródło: Opracowanie własne

3.4.8 MainWindow

MainWindow jest to klasa odpowiadają za okno naszej aplikacji. W niej obsługiwana jest nawigacja oraz wyświetlanie odpowiedniego widoku użytkownikowi.

Plik ten znajduje się w głównym katalogu projektu.



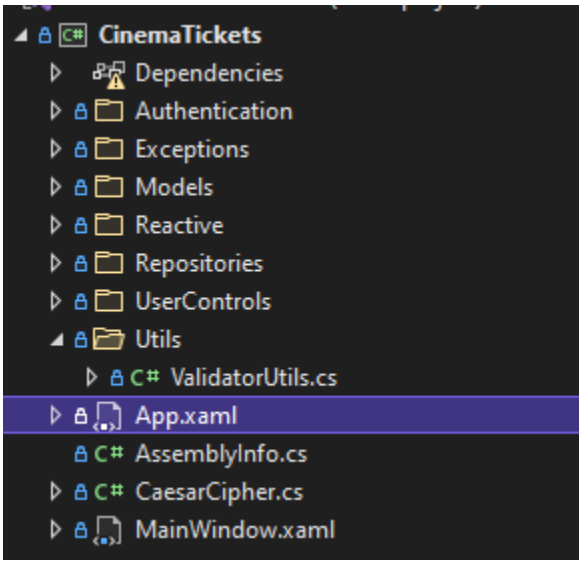
Rysunek 21: Lokalizacja pliku *MainWindow*

Źródło: Opracowanie własne

3.4.9 App

Plik *App* jest plikiem w którym jest skonfigurowane wstrzykiwanie zależności (ang. *Dependency Injection*) oraz połączenie z dostępem do bazy.

Plik ten znajduje się w głównym katalogu projektu.



Rysunek 22: Lokalizacja pliku *App*

Źródło: Opracowanie własne

Dependency Injection realizuje wzorzec strategii jak i sam jest wzorcem projektowym.

3.4.10 Cyklicze usuwanie danych

Cykliczne usuwanie danych zostało zrealizowane przy pomocy pakietu *System.Reactive*. Poniżej znajduje się implementacja:

```
private void InitClearScheduler()
    {
        Observable.Interval(TimeSpan.FromSeconds(20)).ExhaustMap(x =>
        {
            DeleteOldSeances();
            return Observable.Return(x);
        }).Subscribe();

    }

    private async void DeleteOldSeances()
    {
        var oldSeances = _context.MoviesHalls.AsNoTracking().ToList()
            .Where(seance => seance.EndTime < DateTime.Now)
            .ToList();
        if (oldSeances.Count == 0)
        {
            return;
        }
        List<ClientsMoviesHall> odClientSeances = new();
```

```
    foreach (var oldSeance in oldSeances)
    {
        List<ClientsMoviesHall> clientsMoviesHalls = _context.ClientsMoviesHalls.AsNoTracking
            ().Where(clientSeance => clientSeance.IdMovie == oldSeance.IdMovie && clientSeance.I

        odClientSeances.AddRange(clientsMoviesHalls);
    }
    _context.RemoveRange(oldSeances);
    _context.RemoveRange(odClientSeances);
    await _context.SaveChangesAsync();
}
```

3.5 Testy

Testy zostały zrealizowane z wykorzystaniem następujących pakietów:

- MSTest.TestAdapter w wersji 2.2.7,
- MSTest.TestFramework w wersji 2.2.7,
- Microsoft.NET.Test.Sdk w wersji 17.0.0.

Wyniki testów przedstawione na poniższym zrzucie:

Test	Duration
CinemaTickets (3)	2,2 sec
CinemaTickets.Tests (3)	2,2 sec
MovieRepositoryTest (3)	2,2 sec
MovieRepository_ShouldAdd_ReturnAdded	2,1 sec
MovieRepository_ShouldRemove_ReturnNothing	80 ms
MovieRepository_ShouldUpdate_ReturnUpdated	47 ms

Rysunek 23: Wyniki testów

Źródło: Opracowanie własne

Przykład kodu dla testu:

```
[TestMethod]
    public async Task MovieRepository_ShouldRemove_ReturnNothing()
    {
        Movie newMovie = new()
        {
            Title = "Test movie",
            Year = DateTime.Now,
            Duration = 120
        };

        var dbMovie = await _movieRepository!.AddMovieAsync(newMovie);
        await _movieRepository.RemoveMovieAsync(dbMovie);
        var movies = _movieRepository.GetMovies();
        Assert.IsFalse(movies.Any(movie => movie.Id == dbMovie.Id));
    }
```

4 Twórca

Autorem aplikacji jak i dokumentacji jest Adam Czerwiński student Wydziału Matematyki Stosowanej St. II sem. II Informatyka.

Kontrola wersji:

Graph	Description	Date	Author	
	Uncommitted changes	14 lis 2021 23:28	*	*
	main origin/main origin/HEAD **tests**	**14 lis 2021 23:26**	**Adam Czerwiński**	**a90fcdd**
	docs imgs	14 lis 2021 22:14	Adam Czerwiński	180574f
	small improvements	14 lis 2021 22:14	Adam Czerwiński	3c28875
	Attend to seance, catch used movies	14 lis 2021 17:07	Adam Czerwiński	f0ce162
	catch duplicate seances, delete old seances	14 lis 2021 15:36	Adam Czerwiński	814bfc4
	Show seances	13 lis 2021 22:52	Adam Czerwiński	c2650f2
	Seance view	13 lis 2021 19:56	Adam Czerwiński	d7b9239
	Manage halls	13 lis 2021 15:09	Adam Czerwiński	4d7fafe
	Changed Ceaser Cipher implementation, Manage movies, manage employees	13 lis 2021 14:31	Adam Czerwiński	5a71696
	rx package, logout functionality	11 lis 2021 20:24	Adam Czerwiński	e25d4f3
	Client an Employee sing in/ sign up user controls	11 lis 2021 19:08	Adam Czerwiński	8ad2a2b
	Added scopes, move files	6 lis 2021 15:59	Adam Czerwiński	f90d5bc
	Repos, Validations, Password encrypter	6 lis 2021 15:48	Adam Czerwiński	538e05a
	Update gitignore	6 lis 2021 3:13	Adam Czerwiński	38d1d4a
	Simple EmployeeRepository	6 lis 2021 3:10	Adam Czerwiński	9fd6af1
	Added DI	6 lis 2021 2:44	Adam Czerwiński	ac67821
	generate model	6 lis 2021 2:28	Adam Czerwiński	e13ac40
	Reinit project	6 lis 2021 2:28	Adam Czerwiński	16df23d
	Database init script	6 lis 2021 2:28	Adam Czerwiński	c16bb2d
	Init WPF app	4 lis 2021 21:32	Adam-Czerwinski	b5dd52a
	Added gitignore	4 lis 2021 20:57	Adam-Czerwinski	0579ac8
	Initial commit	4 lis 2021 20:56	Adam-Czerwinski	1dbacc0

Rysunek 24: Kontrola wersji

Źródło: Opracowanie własne + SourceTree

5 Kod źródłowy

```
    using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;

namespace CinemaTickets.Utils
{
    public class ValidatorUtils
    {
        private ValidatorUtils()
        {

        }

        public static bool IsValidPassword(string password)
        {
            if (string.IsNullOrWhiteSpace(password))
            {
                return false;
            }

            if (password.Contains(' '))
            {
                return false;
            }

            if (password.Length < 7 || password.Length > 254)
            {
                return false;
            }

            return true;
        }

        public static bool IsValidLogin(string login)
        {
            if (string.IsNullOrWhiteSpace(login))
            {
                return false;
            }

            if (login.Contains(' '))
            {
                return false;
            }

            if (login.Length < 4 || login.Length > 18)
            {
                return false;
            }
```

```
            return true;
        }

        public static bool IsNumber(string text)
        {
            Regex regex = new("[^0-9]+");
            return regex.IsMatch(text);
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CinemaTickets.Authentication
{
    public interface IPasswordCryption
    {
        string EncryptPassword(string password);

        string DecryptPassword(string password);

        bool Matches(string raw, string encrypted);
    }
}

using CinemaTickets.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CinemaTickets.Authentication
{
    public interface IAuthStore
    {
        string? Login { get; }
        AccountType? Type { get; }
        long? Id { get; }
        void Store(AccountType type, string login, long id);

        bool IsLogged();

        void Logout();
    }
}

using CinemaTickets.Models;
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CinemaTickets.Authentication
{
    public interface IAuthService
    {
        long? Authenticate(AccountType type, string login, string password);
    }
}

using CinemaTickets.Models;
using CinemaTickets.Repositories;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CinemaTickets.Authentication
{
    public class AuthService : IAuthService
    {
        private readonly IPasswordCryption _passwordCryption;
        private readonly IEmployeeRepository _employeeRepository;
        private readonly IClientRepository _clientRepository;

        public AuthService(IPasswordCryption passwordCryption, IEmployeeRepository employeeRepository, I
        {
            _passwordCryption = passwordCryption;
            _employeeRepository = employeeRepository;
            _clientRepository = clientRepository;
        }
        public long? Authenticate(AccountType type, string login, string password)
        {
            if (AccountType.CLIENT == type)
            {
                return AuthenticateClient(login, password);
            }
            else
            {
                return AuthenticateEmployee(login, password);
            }
        }

        private long? AuthenticateEmployee(string login, string password)
        {
            if (!_employeeRepository.ExistsByLoginIgnoreCase(login))
            {
                return null;
            }
```

```
            Employee? employee = _employeeRepository.GetByLogin(login);
            if (employee is null)
            {
                return null;
            }

            if (_passwordCryption.Matches(password, employee.Password))
            {
                return employee.Id;
            }
            else
            {
                return null;
            }
        }

        private long? AuthenticateClient(string login, string password)
        {
            if (!_clientRepository.ExistsByLoginIgnoreCase(login))
            {
                return null;
            }

            Client? client = _clientRepository.GetByLogin(login);
            if (client is null)
            {
                return null;
            }

            if (_passwordCryption.Matches(password, client.Password))
            {
                return client.Id;
            }
            else
            {
                return null;
            }
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CinemaTickets.Authentication
{
    public class PasswordCryption : IPasswordCryption
    {
        private readonly int key = 12;

        public string DecryptPassword(string password)
```

```
        {
            return CaesarCipher.Decrypt(password.ToCharArray(), key);
        }

        public string EncryptPassword(string password)
        {
            return CaesarCipher.Encrypt(password.ToCharArray(), key);
        }

        public bool Matches(string raw, string encrypted)
        {
            return string.Equals(EncryptPassword(raw), encrypted);
        }
    }
}


using CinemaTickets.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CinemaTickets.Authentication
{
    public class AuthStore : IAuthStore
    {
        public string? Login { get; private set; }
        public AccountType? Type { get; private set; }

        public long? Id { get; private set; }

        public bool IsLogged()
        {
            return Login != null;
        }

        public void Logout()
        {
            Login = null;
            Type = null;
        }

        public void Store(AccountType type, string login, long id)
        {
            Login = login;
            Type = type;
            Id = id;
        }
    }
}

using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.Text;
using System.Threading.Tasks;

namespace CinemaTickets.Exceptions
{
    [Serializable]
    public class ValidationException : Exception
    {
        public ValidationException()
        {
        }

        public ValidationException(string? message) : base(message)
        {
        }

        public ValidationException(string? message, Exception? innerException) : base(message, innerExce
        {
        }

        protected ValidationException(SerializationInfo info, StreamingContext context) : base(info, co
        {
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.Text;
using System.Threading.Tasks;

namespace CinemaTickets.Exceptions
{
    [Serializable]
    public class NotLoggedException : Exception
    {

        public NotLoggedException()
        {
        }

        public NotLoggedException(string? message) : base(message)
        {
        }

        public NotLoggedException(string? message, Exception? innerException) : base(message, innerExcep
        {
        }
```

```
        protected NotLoggedException(SerializationInfo info, StreamingContext context) : base(info, cont
        {
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CinemaTickets.Exceptions
{
    [Serializable]
    internal class UnexpectedValidationException : Exception
    {
    }
}

using System;
using System.Collections.Generic;

#nullable disable

namespace CinemaTickets.Models
{
    public partial class ClientsMoviesHall
    {
        public long IdClient { get; set; }
        public long IdMovie { get; set; }
        public long IdHall { get; set; }

        public virtual Client IdClientNavigation { get; set; }
        public virtual Hall IdHallNavigation { get; set; }
        public virtual Movie IdMovieNavigation { get; set; }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CinemaTickets.Models
{
    public enum AccountType
    {
        EMPLOYEE,
        CLIENT
    }
}
```

```
using CinemaTickets.Utils;
using System;
using System.Collections.Generic;

#nullable disable

namespace CinemaTickets.Models
{
    public partial class Employee
    {
        public long? Id { get; set; }
        public string Name { get; set; }
        public string LastName { get; set; }
        public string Login { get; set; }
        public string Password { get; set; }

        public bool IsValid(bool validatePassword = false)
        {
            if (validatePassword && !ValidatorUtils.IsValidPassword(Password))
            {
                return false;
            }

            return !string.IsNullOrWhiteSpace(Name)
                && !string.IsNullOrWhiteSpace(LastName)
                && ValidatorUtils.IsValidLogin(Login);
        }
    }
}

using System;
using System.Collections.Generic;

#nullable disable

namespace CinemaTickets.Models
{
    public partial class Hall
    {
        public Hall()
        {
            ClientsMoviesHalls = new HashSet<ClientsMoviesHall>();
            MoviesHalls = new HashSet<MoviesHall>();
        }

        public long? Id { get; set; }
        public int RoomNumber { get; set; }
        public int Size { get; set; }

        public virtual ICollection<ClientsMoviesHall> ClientsMoviesHalls { get; set; }
        public virtual ICollection<MoviesHall> MoviesHalls { get; set; }

        internal bool IsValid()
        {
```

```
            return RoomNumber > 0 && Size > 0;
        }

        public override string ToString()
        {
            return $"Room: {RoomNumber}, Size: {Size}";
        }
    }
}

using System;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata;

#nullable disable

namespace CinemaTickets.Models
{
    public partial class CinematicketsContext : DbContext
    {
        public CinematicketsContext()
        {
        }

        public CinematicketsContext(DbContextOptions<CinematicketsContext> options)
            : base(options)
        {
        }

        public virtual DbSet<Client> Clients { get; set; }
        public virtual DbSet<ClientsMoviesHall> ClientsMoviesHalls { get; set; }
        public virtual DbSet<Employee> Employees { get; set; }
        public virtual DbSet<Hall> Halls { get; set; }
        public virtual DbSet<Movie> Movies { get; set; }
        public virtual DbSet<MoviesHall> MoviesHalls { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            if (!optionsBuilder.IsConfigured)
            {

            }
        }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.HasAnnotation("Relational:Collation", "SQL_Latin1_General_CP1_CI_AS");

            modelBuilder.Entity<Client>(entity =>
            {
                entity.ToTable("clients");

                entity.HasIndex(e => e.Login, "UQ__clients__7838F272B7B79CD7")
                    .IsUnique();
```

```
    entity.Property(e => e.Id).HasColumnName("id");

    entity.Property(e => e.Login)
        .IsRequired()
        .HasMaxLength(18)
        .IsUnicode(false)
        .HasColumnName("login");

    entity.Property(e => e.Password)
        .IsRequired()
        .HasMaxLength(255)
        .IsUnicode(false)
        .HasColumnName("password");
});

modelBuilder.Entity<ClientsMoviesHall>(entity =>
{
    entity.HasKey(e => new { e.IdClient, e.IdMovie, e.IdHall })
        .HasName("PK_CLIENTS_MOVIES_HALLS");

    entity.ToTable("clients_movies_halls");

    entity.Property(e => e.IdClient).HasColumnName("id_client");

    entity.Property(e => e.IdMovie).HasColumnName("id_movie");

    entity.Property(e => e.IdHall).HasColumnName("id_hall");

    entity.HasOne(d => d.IdClientNavigation)
        .WithMany(p => p.ClientsMoviesHalls)
        .HasForeignKey(d => d.IdClient)
        .OnDelete(DeleteBehavior.ClientSetNull)
        .HasConstraintName("clients_movies_halls_fk0");

    entity.HasOne(d => d.IdHallNavigation)
        .WithMany(p => p.ClientsMoviesHalls)
        .HasForeignKey(d => d.IdHall)
        .OnDelete(DeleteBehavior.ClientSetNull)
        .HasConstraintName("clients_movies_halls_fk2");

    entity.HasOne(d => d.IdMovieNavigation)
        .WithMany(p => p.ClientsMoviesHalls)
        .HasForeignKey(d => d.IdMovie)
        .OnDelete(DeleteBehavior.ClientSetNull)
        .HasConstraintName("clients_movies_halls_fk1");
});

modelBuilder.Entity<Employee>(entity =>
{
    entity.ToTable("employees");

    entity.HasIndex(e => e.Login, "UQ__employee__7838F272D2760A53")
        .IsUnique();
```

```
    entity.Property(e => e.Id).HasColumnName("id");

    entity.Property(e => e.LastName)
        .IsRequired()
        .HasMaxLength(50)
        .IsUnicode(false)
        .HasColumnName("last_name");

    entity.Property(e => e.Login)
        .IsRequired()
        .HasMaxLength(18)
        .IsUnicode(false)
        .HasColumnName("login");

    entity.Property(e => e.Name)
        .IsRequired()
        .HasMaxLength(50)
        .IsUnicode(false)
        .HasColumnName("name");

    entity.Property(e => e.Password)
        .IsRequired()
        .HasMaxLength(255)
        .IsUnicode(false)
        .HasColumnName("password");
});

modelBuilder.Entity<Hall>(entity =>
{
    entity.ToTable("halls");

    entity.Property(e => e.Id).HasColumnName("id");

    entity.Property(e => e.RoomNumber).HasColumnName("room_number");

    entity.Property(e => e.Size).HasColumnName("size");
});

modelBuilder.Entity<Movie>(entity =>
{
    entity.ToTable("movies");

    entity.Property(e => e.Id).HasColumnName("id");

    entity.Property(e => e.Duration).HasColumnName("duration");

    entity.Property(e => e.Title)
        .IsRequired()
        .HasMaxLength(255)
        .IsUnicode(false)
        .HasColumnName("title");

    entity.Property(e => e.Year)
```

```
                .HasColumnType("date")
                .HasColumnName("year");
            });

            modelBuilder.Entity<MoviesHall>(entity =>
            {
                entity.HasKey(e => new { e.IdMovie, e.IdHall })
                    .HasName("PK_MOVIES_HALLS");

                entity.ToTable("movies_halls");

                entity.Property(e => e.IdMovie).HasColumnName("id_movie");

                entity.Property(e => e.IdHall).HasColumnName("id_hall");

                entity.Property(e => e.EndTime)
                    .HasColumnType("datetime")
                    .HasColumnName("end_time");

                entity.Property(e => e.StartTime)
                    .HasColumnType("datetime")
                    .HasColumnName("start_time");

                entity.HasOne(d => d.IdHallNavigation)
                    .WithMany(p => p.MoviesHalls)
                    .HasForeignKey(d => d.IdHall)
                    .OnDelete(DeleteBehavior.ClientSetNull)
                    .HasConstraintName("movies_halls_fk1");

                entity.HasOne(d => d.IdMovieNavigation)
                    .WithMany(p => p.MoviesHalls)
                    .HasForeignKey(d => d.IdMovie)
                    .OnDelete(DeleteBehavior.ClientSetNull)
                    .HasConstraintName("movies_halls_fk0");
            });

            OnModelCreatingPartial(modelBuilder);
        }

        partial void OnModelCreatingPartial(ModelBuilder modelBuilder);
    }
}

using System;
using System.Collections.Generic;

#nullable disable

namespace CinemaTickets.Models
{
    public partial class Movie
    {
        public Movie()
        {
```

```
            ClientsMoviesHalls = new HashSet<ClientsMoviesHall>();
            MoviesHalls = new HashSet<MoviesHall>();
        }

        public long? Id { get; set; }
        public string Title { get; set; }
        public DateTime Year { get; set; }
        public int Duration { get; set; }

        public virtual ICollection<ClientsMoviesHall> ClientsMoviesHalls { get; set; }
        public virtual ICollection<MoviesHall> MoviesHalls { get; set; }

        public bool IsValid()
        {
            return !string.IsNullOrWhiteSpace(Title)
                && Duration is > 0 &&
                Year != new DateTime();
        }

        public override string ToString()
        {
            return $"{Title} ({Year.Year})";
        }
    }
}

using System;
using System.Collections.Generic;

#nullable disable

namespace CinemaTickets.Models
{
    public partial class MoviesHall
    {
        public long IdMovie { get; set; }
        public long IdHall { get; set; }
        public DateTime StartTime { get; set; }
        public DateTime EndTime { get; set; }

        public virtual Hall IdHallNavigation { get; set; }
        public virtual Movie IdMovieNavigation { get; set; }
    }
}

using System;
using System.Collections.Generic;

#nullable disable

namespace CinemaTickets.Models
{
    public partial class Client
    {
```

```
        public Client()
        {
            ClientsMoviesHalls = new HashSet<ClientsMoviesHall>();
        }

        public long? Id { get; set; }
        public string Login { get; set; }
        public string Password { get; set; }

        public virtual ICollection<ClientsMoviesHall> ClientsMoviesHalls { get; set; }
    }
}

using CinemaTickets.Models;
using CinemaTickets.UserControls.Employees;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reactive.Subjects;
using System.Text;
using System.Threading.Tasks;

namespace CinemaTickets.Reactive
{
    public class EmployeeReactiveUtils
    {
        private EmployeeReactiveUtils() { }

        private static readonly Subject<SingleEmployeeUserControl> EditEmployeeSubject = new();
        public static readonly IObservable<SingleEmployeeUserControl> EditEmployeeObservable = EditEmplo

        private static readonly Subject<SingleEmployeeUserControl> CancelEditEmployeeSubject = new();
        public static readonly IObservable<SingleEmployeeUserControl> CancelEditEmployeeObservable = Can

        private static readonly Subject<long> DeleteEmployeeSubject = new();
        public static readonly IObservable<long> DeleteEmployeeObservable = DeleteEmployeeSubject;

        private static readonly Subject<Employee> SaveEmployeeSubject = new();
        public static readonly IObservable<Employee> SaveEmployeeObservable = SaveEmployeeSubject;

        public static void OnEditEmployee(SingleEmployeeUserControl singleEmployeeUserControl)
        {
            EditEmployeeSubject.OnNext(singleEmployeeUserControl);
        }

        public static void OnCancelEditEmployee(SingleEmployeeUserControl singleEmployeeUserControl)
        {
            CancelEditEmployeeSubject.OnNext(singleEmployeeUserControl);
        }
        public static void OnDeleteEmployee(long id)
        {
            DeleteEmployeeSubject.OnNext(id);
        }
```

```
        public static void OnSaveEmployee(Employee Employee)
        {
            SaveEmployeeSubject.OnNext(Employee);
        }
    }
}

using CinemaTickets.Models;
using CinemaTickets.UserControls.Halls;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reactive.Subjects;
using System.Text;
using System.Threading.Tasks;

namespace CinemaTickets.Reactive
{

    public class HallReactiveUtils
    {
        private static readonly Subject<SingleHallUserControl> EditHallSubject = new();
        public static readonly IObservable<SingleHallUserControl> EditHallObservable = EditHallSubject;

        private static readonly Subject<SingleHallUserControl> CancelEditHallSubject = new();
        public static readonly IObservable<SingleHallUserControl> CancelEditHallObservable = CancelEditH

        private static readonly Subject<Hall> SaveHallSubject = new();
        public static readonly IObservable<Hall> SaveHallObservable = SaveHallSubject;

        private HallReactiveUtils() { }

        public static void OnEditHall(SingleHallUserControl singleHallUserControl)
        {
            EditHallSubject.OnNext(singleHallUserControl);
        }

        public static void OnCancelEditHall(SingleHallUserControl singleHallUserControl)
        {
            CancelEditHallSubject.OnNext(singleHallUserControl);
        }

        public static void OnSaveHall(Hall Hall)
        {
            SaveHallSubject.OnNext(Hall);
        }
    }
}

using CinemaTickets.Models;
using CinemaTickets.UserControls.Seances;
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Reactive.Subjects;
using System.Text;
using System.Threading.Tasks;

namespace CinemaTickets.Reactive
{
    public class SeanceReactiveUtils
    {
        private static readonly Subject<MoviesHall> SaveSeanceSubject = new();
        public static readonly IObservable<MoviesHall> SaveSeanceObservable = SaveSeanceSubject;

        private static readonly Subject<NewSeanceUserControl> CancelEditSeanceSubject = new();
        public static readonly IObservable<NewSeanceUserControl> CancelEditSeanceObservable = CancelEdit


        private static readonly Subject<MoviesHall> AttendSeanceSubject = new();
        public static readonly IObservable<MoviesHall> AttendSeanceObservable = AttendSeanceSubject;

        private static readonly Subject<MoviesHall> CancelAttendSeanceSubject = new();
        public static readonly IObservable<MoviesHall> CancelAttendSeanceObservable = CancelAttendSeance
        private SeanceReactiveUtils()
        {
        }

        public static void OnCancelEditSeance(NewSeanceUserControl newSeanceUserControl)
        {
            CancelEditSeanceSubject.OnNext(newSeanceUserControl);
        }

        public static void OnSaveSeance(MoviesHall seance)
        {
            SaveSeanceSubject.OnNext(seance);
        }

        public static void OnAttendSeance(MoviesHall seance)
        {
            AttendSeanceSubject.OnNext(seance);
        }

        public static void OnCancelAttendSeance(MoviesHall seance)
        {
            CancelAttendSeanceSubject.OnNext(seance);
        }
    }
}

using CinemaTickets.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reactive.Subjects;
using System.Text;
using System.Threading.Tasks;
```

```
namespace CinemaTickets.Reactive
{
    public class LoginReactiveUtils
    {
        private static readonly Subject<AccountType> LoginSubject = new();
        public static readonly IObservable<AccountType> LoginObservable = LoginSubject;

        private LoginReactiveUtils()
        {
        }

        public static void OnLogin(AccountType accountType)
        {
            LoginSubject.OnNext(accountType);
        }
    }
}

using System;
using System.Collections.Generic;
using System.Reactive.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace CinemaTickets.Reactive
{
    public static class CustomReactiveOperators
    {
        /// <summary>
        /// Projects each element to an observable sequence, which is merged
        /// in the output observable sequence only if the previous projected observable
        /// sequence has completed.
        /// source: https://stackoverflow.com/a/64356119
        /// </summary>
        public static IObservable<TResult> ExhaustMap<TSource, TResult>(
            this IObservable<TSource> source,
            Func<TSource, IObservable<TResult>> function)
        {
            return Observable.Using(() => new SemaphoreSlim(1, 1),
                semaphore => source.SelectMany(item => ProjectItem(item, semaphore)));

            IObservable<TResult> ProjectItem(TSource item, SemaphoreSlim semaphore)
            {
                // Attempt to acquire the semaphore immediately. If successful, return
                // a sequence that releases the semaphore when terminated. Otherwise,
                // return immediately an empty sequence.
                return Observable.If(() => semaphore.Wait(0),
                    Observable
                        .Defer(() => function(item))
                        .Finally(() => semaphore.Release())
                );
            }
        }
```

```
    }
}

using CinemaTickets.Models;
using CinemaTickets.UserControls.Movies;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reactive.Subjects;
using System.Text;
using System.Threading.Tasks;

namespace CinemaTickets.Reactive
{
    class MovieReactiveUtils
    {
        private static readonly Subject<SingleMovieUserControl> EditMovieSubject = new();
        public static readonly IObservable<SingleMovieUserControl> EditMovieObservable = EditMovieSubjec

        private static readonly Subject<SingleMovieUserControl> CancelEditMovieSubject = new();
        public static readonly IObservable<SingleMovieUserControl> CancelEditMovieObservable = CancelEdi

        private static readonly Subject<long> DeleteMovieSubject = new();
        public static readonly IObservable<long> DeleteMovieObservable = DeleteMovieSubject;

        private static readonly Subject<Movie> SaveMovieSubject = new();
        public static readonly IObservable<Movie> SaveMovieObservable = SaveMovieSubject;
        private MovieReactiveUtils()
        {

        }

        public static void OnEditMovie(SingleMovieUserControl singleMovieUserControl)
        {
            EditMovieSubject.OnNext(singleMovieUserControl);
        }

        public static void OnCancelEditMovie(SingleMovieUserControl singleMovieUserControl)
        {
            CancelEditMovieSubject.OnNext(singleMovieUserControl);
        }
        public static void OnDeleteMovie(long id)
        {
            DeleteMovieSubject.OnNext(id);
        }

        public static void OnSaveMovie(Movie movie)
        {
            SaveMovieSubject.OnNext(movie);
        }
    }
}

using CinemaTickets.Exceptions;
```

```
using CinemaTickets.Models;
using CinemaTickets.Utils;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;
namespace CinemaTickets.Repositories
{
    public class ClientRepository : IClientRepository
    {
        private readonly CinematicketsContext _context;

        public ClientRepository(CinematicketsContext context)
        {
            _context = context;
        }

        public async Task<Client> AddClientAsync(Client client)
        {
            if (!ValidatorUtils.IsValidLogin(client.Login))
            {
                throw new ValidationException("Nieprawidłowy login");
            }
            _context.Clients.Add(client);
            await _context.SaveChangesAsync();
            _context.Entry(client).State = EntityState.Detached;
            return client;
        }

        public bool ExistsByLoginIgnoreCase(string login)
        {
            return _context.Clients.Any(client => client.Login.ToLower() == login.ToLower());
        }

        public Client? GetByLogin(string login)
        {
            return _context.Clients.AsNoTracking().SingleOrDefault(client => login.Equals(client.Login))
        }
    }
}

using CinemaTickets.Exceptions;
using CinemaTickets.Models;
using CinemaTickets.Utils;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;
namespace CinemaTickets.Repositories
{
```

```
    public class EmployeeRepository : IEmployeeRepository
    {
        private readonly CinematicketsContext _context;

        public EmployeeRepository(CinematicketsContext context)
        {
            _context = context;
        }

        public async Task<Employee> AddEmployeeAsync(Employee employee)
        {
            if (!ValidatorUtils.IsValidLogin(employee.Login))
            {
                throw new ValidationException("Nieprawidłowy login");
            }
            _context.Employees.Add(employee);
            await _context.SaveChangesAsync();
            _context.Entry(employee).State = EntityState.Detached;
            return employee;
        }

        public bool ExistsByLoginIgnoreCase(string login)
        {
            return _context.Employees.Any(employee => employee.Login.ToLower() == login.ToLower());
        }

        public Employee? GetByLogin(string login)
        {
            return _context.Employees.AsNoTracking().SingleOrDefault(employee => login.Equals(employee.L
        }

        public List<Employee> GetEmployees()
        {
            return _context.Employees.AsNoTracking().ToList();
        }

        public async Task RemoveEmployeeAsync(Employee employee)
        {
            _context.Employees.Remove(employee);
            await _context.SaveChangesAsync();
        }

        public async Task<Employee> UpdateEmployeeAsync(Employee employee)
        {
            _context.Employees.Update(employee);
            await _context.SaveChangesAsync();
            _context.Entry(employee).State = EntityState.Detached;
            return employee;
        }
    }
}

using CinemaTickets.Models;
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;
namespace CinemaTickets.Repositories
{
    public class HallRepository : IHallRepository
    {
        private readonly CinematicketsContext _context;

        public HallRepository(CinematicketsContext context)
        {
            _context = context;
        }
        public async Task<Hall> AddHallAsync(Hall hall)
        {
            _context.Halls.Add(hall);
            await _context.SaveChangesAsync();
            _context.Entry(hall).State = EntityState.Detached;
            return hall;
        }

        public bool ExistsByRoomNumber(int number)
        {
            return _context.Halls.Any(hall => hall.RoomNumber == number);
        }

        public Hall? GetHallById(long? id)
        {
            return _context.Halls.AsNoTracking().SingleOrDefault(hall => hall.Id == id);
        }

        public List<Hall> GetHalls()
        {
            return _context.Halls.AsNoTracking().ToList();
        }

        public async Task RemoveHallAsync(Hall hall)
        {
            _context.Halls.Remove(hall);
            await _context.SaveChangesAsync();
        }

        public async Task<Hall> UpdateHallAsync(Hall hall)
        {
            _context.Halls.Update(hall);
            await _context.SaveChangesAsync();
            _context.Entry(hall).State = EntityState.Detached;
            return hall;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CinemaTickets.Repositories
{
    public interface IDetachable
    {
        void Detach(object entity);
    }
}

using CinemaTickets.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CinemaTickets.Repositories
{
    public interface IClientRepository
    {
        Task<Client> AddClientAsync(Client client);

        bool ExistsByLoginIgnoreCase(string login);

        Client? GetByLogin(string login);
    }
}

using CinemaTickets.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CinemaTickets.Repositories
{
    public interface IEmployeeRepository
    {
        Task<Employee> AddEmployeeAsync(Employee employee);

        Task<Employee> UpdateEmployeeAsync(Employee employee);

        Task RemoveEmployeeAsync(Employee employee);

        List<Employee> GetEmployees();

        bool ExistsByLoginIgnoreCase(string login);
```

```
        Employee? GetByLogin(string login);
    }
}

using CinemaTickets.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CinemaTickets.Repositories
{
    public interface IHallRepository
    {
        Task<Hall> AddHallAsync(Hall hall);

        Task<Hall> UpdateHallAsync(Hall hall);

        Task RemoveHallAsync(Hall hall);

        List<Hall> GetHalls();

        bool ExistsByRoomNumber(int number);
        Hall? GetHallById(long? id);
    }
}

using CinemaTickets.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CinemaTickets.Repositories
{
    public interface IMovieRepository : IDetachable
    {
        Task<Movie> AddMovieAsync(Movie movie);

        Task<Movie> UpdateMovieAsync(Movie movie);

        Task RemoveMovieAsync(Movie movie);

        List<Movie> GetMovies();

        bool IsMovieUsed(long id);
    }
}

using CinemaTickets.Models;
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CinemaTickets.Repositories
{
    public interface ISeanceRegistrationRepository
    {
        Task<ClientsMoviesHall> AttendAsync(ClientsMoviesHall clientsMoviesHall);
        List<ClientsMoviesHall> GetSeances(long clientId);
        Task RemoveAttendAsync(ClientsMoviesHall clientsMoviesHall);
    }
}

using CinemaTickets.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CinemaTickets.Repositories
{
    public interface ISeanceRepository : IDetachable
    {
        Task<MoviesHall> AddSeanceAsync(MoviesHall moviesHall);

        Task<MoviesHall> UpdateSeanceAsync(MoviesHall moviesHall);

        Task RemoveSeanceAsync(MoviesHall moviesHall);

        List<MoviesHall> GetSeances();
    }
}

using CinemaTickets.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;
namespace CinemaTickets.Repositories
{
    internal class MovieRepository : IMovieRepository
    {
        private readonly CinematicketsContext _context;

        public MovieRepository(CinematicketsContext context)
        {
            _context = context;
        }
        public async Task<Movie> AddMovieAsync(Movie movie)
        {
```

```
            _context.Movies.Add(movie);
            await _context.SaveChangesAsync();
            _context.Entry(movie).State = EntityState.Detached;
            return movie;
        }

        public void Detach(object entity)
        {
            _context.Entry(entity).State = EntityState.Detached;
        }

        public List<Movie> GetMovies()
        {
            return _context.Movies
                .AsNoTracking()
                .ToList();
        }

        public async Task RemoveMovieAsync(Movie movie)
        {
            _context.Movies.Remove(movie);
            await _context.SaveChangesAsync();
        }

        public bool IsMovieUsed(long id)
        {
            return _context.MoviesHalls.AsNoTracking().Any(seance => seance.IdMovie == id);
        }

        public async Task<Movie> UpdateMovieAsync(Movie movie)
        {
            _context.Movies.Update(movie);
            await _context.SaveChangesAsync();
            _context.Entry(movie).State = EntityState.Detached;
            return movie;
        }
    }
}

using CinemaTickets.Models;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CinemaTickets.Repositories
{
    public class SeanceRegistrationRepository : ISeanceRegistrationRepository
    {
        private readonly CinematicketsContext _context;

        public SeanceRegistrationRepository(CinematicketsContext context)
```

```
        {
            _context = context;
        }

        public async Task<ClientsMoviesHall> AttendAsync(ClientsMoviesHall clientsMoviesHall)
        {
            _context.ClientsMoviesHalls.Add(clientsMoviesHall);
            await _context.SaveChangesAsync();
            _context.Entry(clientsMoviesHall).State = EntityState.Detached;
            return clientsMoviesHall;
        }

        public List<ClientsMoviesHall> GetSeances(long clientId)
        {
            return _context.ClientsMoviesHalls.AsNoTracking().Where(seances => seances.IdClient == clier
        }

        public async Task RemoveAttendAsync(ClientsMoviesHall clientsMoviesHall)
        {
            _context.ClientsMoviesHalls.Remove(clientsMoviesHall);
            await _context.SaveChangesAsync();
        }
    }
}

using CinemaTickets.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;
using System.Reactive.Linq;
using CinemaTickets.Reactive;

namespace CinemaTickets.Repositories
{
    public class SeanceRepository : ISeanceRepository
    {
        private readonly CinematicketsContext _context;

        public SeanceRepository(CinematicketsContext context)
        {
            _context = context;
            InitClearScheduler();
        }

        private void InitClearScheduler()
        {
            Observable.Interval(TimeSpan.FromSeconds(20)).ExhaustMap(x =>
            {
                DeleteOldSeances();
                return Observable.Return(x);
            }).Subscribe();
```

```
}

private async void DeleteOldSeances()
{
    var oldSeances = _context.MoviesHalls.AsNoTracking().ToList()
        .Where(seance => seance.EndTime < DateTime.Now)
        .ToList();
    if (oldSeances.Count == 0)
    {
        return;
    }
    List<ClientsMoviesHall> odClientSeances = new();
    foreach (var oldSeance in oldSeances)
    {
        List<ClientsMoviesHall> clientsMoviesHalls = _context.ClientsMoviesHalls.AsNoTracking
            ().Where(clientSeance => clientSeance.IdMovie == oldSeance.IdMovie && clientSeance.I

        odClientSeances.AddRange(clientsMoviesHalls);
    }
    _context.RemoveRange(oldSeances);
    _context.RemoveRange(odClientSeances);
    await _context.SaveChangesAsync();
}

public async Task<MoviesHall> AddSeanceAsync(MoviesHall moviesHall)
{
    _context.MoviesHalls.Add(moviesHall);
    await _context.SaveChangesAsync();
    _context.Entry(moviesHall).State = EntityState.Detached;
    return moviesHall;
}

public void Detach(object entity)
{
    _context.Entry(entity).State = EntityState.Detached;
}

public List<MoviesHall> GetSeances()
{
    return _context.MoviesHalls.AsNoTracking()
        .Include(seance => seance.IdHallNavigation)
        .AsNoTracking()
        .Include(seance => seance.IdMovieNavigation)
        .AsNoTracking()
        .ToList().Where(seance => seance.EndTime >= DateTime.Now).ToList();
}

public async Task RemoveSeanceAsync(MoviesHall moviesHall)
{
    _context.MoviesHalls.Remove(moviesHall);
    await _context.SaveChangesAsync();
}
```

```
        public async Task<MoviesHall> UpdateSeanceAsync(MoviesHall moviesHall)
        {
            _context.MoviesHalls.Update(moviesHall);
            await _context.SaveChangesAsync();
            _context.Entry(moviesHall).State = EntityState.Detached;
            return moviesHall;
        }
    }
}

using CinemaTickets.Models;
using CinemaTickets.Repositories;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CinemaTickets.Tests
{
    [TestClass]
    public class MovieRepositoryTest
    {
        private ServiceProvider? serviceProvider;
        private IMovieRepository? _movieRepository;

        [TestInitialize]
        public void Initialize()
        {
            ServiceCollection services = new();
            ConfigureServices(services);
            serviceProvider = services.BuildServiceProvider();
            _movieRepository = serviceProvider.GetService<IMovieRepository>();
        }

        private void ConfigureServices(ServiceCollection services)
        {
            services.AddDbContext<CinematicketsContext>(options => options.UseSqlServer("Data Source=loc
            services.AddScoped<IMovieRepository, MovieRepository>();
        }


        [TestMethod]
        public async Task MovieRepository_ShouldAdd_ReturnAdded()
        {
            Movie newMovie = new()
            {
                Title = "Test movie",
                Year = DateTime.Now,
                Duration = 120
            };
```

```
            var dbMovie = await _movieRepository!.AddMovieAsync(newMovie);

            Assert.AreEqual(newMovie.Title, dbMovie.Title);
            Assert.AreEqual(newMovie.Year, dbMovie.Year);
            Assert.AreEqual(newMovie.Duration, dbMovie.Duration);
            var movies = _movieRepository.GetMovies();
            Assert.IsTrue(movies.Any(movie => movie.Title == newMovie.Title && movie.Id == newMovie.Id))
        }

        [TestMethod]
        public async Task MovieRepository_ShouldUpdate_ReturnUpdated()
        {
            Movie newMovie = new()
            {
                Title = "Test movie",
                Year = DateTime.Now,
                Duration = 120
            };

            var dbMovie = await _movieRepository!.AddMovieAsync(newMovie);

            dbMovie.Title = "Update title";
            var updatedDbMovie = await _movieRepository!.UpdateMovieAsync(dbMovie);
            var movies = _movieRepository.GetMovies();
            Assert.IsTrue(movies.Any(movie => movie.Title == updatedDbMovie.Title && movie.Id == updated
        }

        [TestMethod]
        public async Task MovieRepository_ShouldRemove_ReturnNothing()
        {
            Movie newMovie = new()
            {
                Title = "Test movie",
                Year = DateTime.Now,
                Duration = 120
            };

            var dbMovie = await _movieRepository!.AddMovieAsync(newMovie);
            await _movieRepository.RemoveMovieAsync(dbMovie);
            var movies = _movieRepository.GetMovies();
            Assert.IsFalse(movies.Any(movie => movie.Id == dbMovie.Id));
        }


    }
}

<UserControl x:Class="CinemaTickets.UserControls.Employees.EmployeesUserControl"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:CinemaTickets.UserControls.Employees"
```

```
            mc:Ignorable="d"
            d:DesignHeight="450" d:DesignWidth="800">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="9*"></RowDefinition>
            <RowDefinition></RowDefinition>
        </Grid.RowDefinitions>
        <WrapPanel Grid.Row="0" x:Name="EmployeesWrapPanel"></WrapPanel>
        <Button x:Name="AddButton" Grid.Row="1" Content="Add" Click="OnAddEmployeeClick"/>
    </Grid>
</UserControl>

<UserControl x:Class="CinemaTickets.UserControls.Employees.SingleEmployeeUserControl"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:CinemaTickets.UserControls.Employees"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800">
    <StackPanel Orientation="Vertical">
        <StackPanel Orientation="Horizontal">
            <Label Content="Login" Width="70"/>
            <TextBox x:Name="EmployeeLoginTextBlock" MinWidth="40" Width="100" TextChanged="OnEmployeeLo
        </StackPanel>
        <StackPanel Orientation="Horizontal">
            <Label Content="Name" Width="70"/>
            <TextBox x:Name="EmployeeNameTextBlock" MinWidth="40" Width="100" TextChanged="OnEmployeeNam
        </StackPanel>
        <StackPanel Orientation="Horizontal">
            <Label Content="Last Name" Width="70"/>
            <TextBox x:Name="EmployeeLastNameTextBlock" MinWidth="40" Width="100" TextChanged="OnEmploye
        </StackPanel>
        <StackPanel x:Name="PasswordStackPanel" Orientation="Horizontal">
            <Label Content="Password" Width="70"/>
            <PasswordBox x:Name="EmployeePasswordPasswordBox" MinWidth="40" Width="100" PasswordChanged=
        </StackPanel>

        <StackPanel Orientation="Horizontal">
            <Button x:Name="EditButton" Content="Edit" Margin="0,10,0,0" Width="170" HorizontalAlignment
            <Button x:Name="CancelButton" Content="Cancel" Margin="0,10,0,0" Width="85" HorizontalAlignm
            <Button x:Name="SaveButton" Content="Save" Margin="0,10,0,0" Width="85" HorizontalAlignment=
        </StackPanel>
    </StackPanel>
</UserControl>

using CinemaTickets.Authentication;
using CinemaTickets.Models;
using CinemaTickets.Reactive;
using CinemaTickets.Repositories;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reactive.Linq;
```

```
using System.Reactive.Subjects;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace CinemaTickets.UserControls.Employees
{
    /// <summary>
    /// Interaction logic for EmployeesUserControl.xaml
    /// </summary>
    public partial class EmployeesUserControl : UserControl, IDisposable
    {
        private readonly Subject<object?> _destroySubject = new();
        private readonly IEmployeeRepository _employeeRepository;
        private readonly IPasswordCryption _passwordCryption;
        private bool disposedValue;

        public EmployeesUserControl(IEmployeeRepository employeeRepository, IPasswordCryption passwordCr
        {
            InitializeComponent();
            _employeeRepository = employeeRepository;
            _passwordCryption = passwordCryption;
            EmployeeReactiveUtils.CancelEditEmployeeObservable.TakeUntil(_destroySubject).Subscribe(cont
            {
                ShowEmployees();
                AddButton.Visibility = Visibility.Visible;
            });

            EmployeeReactiveUtils.EditEmployeeObservable.TakeUntil(_destroySubject).Subscribe(control =>
            {
                foreach (SingleEmployeeUserControl employeeControl in EmployeesWrapPanel.Children)
                {
                    if (employeeControl == control)
                    {
                        continue;
                    }

                    employeeControl.HideButtons();
                }
                AddButton.Visibility = Visibility.Hidden;
            });

            EmployeeReactiveUtils.SaveEmployeeObservable.TakeUntil(_destroySubject).Subscribe(async empl
            {
                if (employee.Id is not null)
                {
```

```
            var updateEmployee = new Employee()
            {
                Id = employee.Id,
                Login = employee.Login,
                Name = employee.Name,
                LastName = employee.LastName,
                Password = _passwordCryption.EncryptPassword(employee.Password),
            };
            await _employeeRepository.UpdateEmployeeAsync(updateEmployee);
            MessageBox.Show("Employee has been updated successfully");
        }
        else
        {
            if (_employeeRepository.ExistsByLoginIgnoreCase(employee.Login))
            {
                MessageBox.Show("Login already in use");
                return;
            }
            var newEmployee = new Employee()
            {
                Login = employee.Login,
                Name = employee.Name,
                LastName = employee.LastName,
                Password = _passwordCryption.EncryptPassword(employee.Password),
            };
            await _employeeRepository.AddEmployeeAsync(newEmployee);
            MessageBox.Show("Employee has been added successfully");
        }
        ShowEmployees();
        AddButton.Visibility = Visibility.Visible;
    });

    ShowEmployees();
}

private void ShowEmployees()
{
    EmployeesWrapPanel.Children.Clear();
    List<Models.Employee> employees = _employeeRepository.GetEmployees();
    foreach (var employee in employees)
    {
        employee.Password = _passwordCryption.DecryptPassword(employee.Password);
    }
    var singleEmployeeUserControls = employees.Select(employee =>
        new SingleEmployeeUserControl(employee)
    ).ToList();
    singleEmployeeUserControls.ForEach(employeeUserControl => EmployeesWrapPanel.Children.Add(em
}

private void OnAddEmployeeClick(object sender, RoutedEventArgs e)
{
    foreach (SingleEmployeeUserControl item in EmployeesWrapPanel.Children)
    {
        item.HideButtons();
```

```
            }
            EmployeesWrapPanel.Children.Add(new SingleEmployeeUserControl(null));
            AddButton.Visibility = Visibility.Hidden;
        }

        protected virtual void Dispose(bool disposing)
        {
            if (!disposedValue)
            {
                if (disposing)
                {
                    // TODO: dispose managed state (managed objects)
                    _destroySubject.OnNext(null);
                }

                // TODO: free unmanaged resources (unmanaged objects) and override finalizer
                // TODO: set large fields to null
                disposedValue = true;
            }
        }

        // // TODO: override finalizer only if 'Dispose(bool disposing)' has code to free unmanaged reso
        // ~EmployeesUserControl()
        // {
        //     // Do not change this code. Put cleanup code in 'Dispose(bool disposing)' method
        //     Dispose(disposing: false);
        // }

        public void Dispose()
        {
            // Do not change this code. Put cleanup code in 'Dispose(bool disposing)' method
            Dispose(disposing: true);
            GC.SuppressFinalize(this);
        }
    }
}

using CinemaTickets.Authentication;
using CinemaTickets.Models;
using CinemaTickets.Reactive;
using CinemaTickets.Utils;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
```

```
using System.Windows.Shapes;

namespace CinemaTickets.UserControls.Employees
{
    /// <summary>
    /// Interaction logic for SingleEmployeeUserControl.xaml
    /// </summary>
    public partial class SingleEmployeeUserControl : UserControl
    {
        private bool _editing = false;
        private readonly bool _new;

        public Employee Employee { get; set; }
        public SingleEmployeeUserControl(Employee? employee)
        {
            InitializeComponent();
            _new = employee is null || employee.Id is null;
            if (employee is not null)
            {
                Employee = employee;
                LoadContent();
                SetEditControls(false);
            }
            else
            {
                Employee = new();
                SetEditControls(true);
            }

        }
        private void LoadContent()
        {
            EmployeeLoginTextBlock.Text = Employee.Login;
            EmployeeNameTextBlock.Text = Employee.Name;
            EmployeeLastNameTextBlock.Text = Employee.LastName;
            EmployeePasswordPasswordBox.Password = Employee.Password;
        }

        private void SetEditControls(bool edit)
        {
            _editing = edit;

            EmployeeLoginTextBlock.IsEnabled = edit && _new;
            EmployeeNameTextBlock.IsEnabled = edit;
            EmployeeLastNameTextBlock.IsEnabled = edit;
            PasswordStackPanel.Visibility = edit ? Visibility.Visible : Visibility.Collapsed;
            EmployeePasswordPasswordBox.IsEnabled = edit;

            SaveButton.Visibility = edit ? Visibility.Visible : Visibility.Collapsed;
            CancelButton.Visibility = edit ? Visibility.Visible : Visibility.Collapsed;
            EditButton.Visibility = !edit ? Visibility.Visible : Visibility.Collapsed;

            SaveButton.IsEnabled = _editing && IsValidEmployee();
        }
```

```
private bool IsValidEmployee()
{
    var tempEmployee = new Employee()
    {
        Login = EmployeeLoginTextBlock.Text.Trim(),
        Name = EmployeeNameTextBlock.Text.Trim(),
        LastName = EmployeeLastNameTextBlock.Text.Trim(),
        Password = EmployeePasswordPasswordBox.Password.Trim()
    };
    return tempEmployee.IsValid(true);
}

public void HideButtons()
{
    SaveButton.Visibility = Visibility.Collapsed;
    CancelButton.Visibility = Visibility.Collapsed;
    EditButton.Visibility = Visibility.Collapsed;
}


private void OnCancelClick(object sender, RoutedEventArgs e)
{
    EmployeeReactiveUtils.OnCancelEditEmployee(this);
}

private void OnEditClick(object sender, RoutedEventArgs e)
{
    SetEditControls(true);
    EmployeeReactiveUtils.OnEditEmployee(this);
}

private void OnSaveClick(object sender, RoutedEventArgs e)
{
    Employee.Login = EmployeeLoginTextBlock.Text.Trim();
    Employee.Name = EmployeeNameTextBlock.Text.Trim();
    Employee.LastName = EmployeeLastNameTextBlock.Text.Trim();
    Employee.Password = EmployeePasswordPasswordBox.Password.Trim();
    EmployeeReactiveUtils.OnSaveEmployee(Employee);
}

private void OnEmployeeLoginTextChanged(object sender, TextChangedEventArgs e)
{
    SaveButton.IsEnabled = IsValidEmployee();
}

private void OnEmployeeNameTextChanged(object sender, TextChangedEventArgs e)
{
    SaveButton.IsEnabled = IsValidEmployee();
}

private void OnEmployeeLastNameTextChanged(object sender, TextChangedEventArgs e)
{
    SaveButton.IsEnabled = IsValidEmployee();
```

```
        }

        private void OnEmployeePasswordChanged(object sender, RoutedEventArgs e)
        {
            SaveButton.IsEnabled = IsValidEmployee();
        }
    }
}

<UserControl x:Class="CinemaTickets.UserControls.Halls.HallsUserControl"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:CinemaTickets.UserControls.Halls"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="9*"></RowDefinition>
            <RowDefinition></RowDefinition>
        </Grid.RowDefinitions>
        <WrapPanel Grid.Row="0" x:Name="HallsWrapPanel"></WrapPanel>
        <Button x:Name="AddButton" Grid.Row="1" Content="Add" Click="OnAddHallClick"/>
    </Grid>
</UserControl>

using CinemaTickets.Models;
using CinemaTickets.Reactive;
using CinemaTickets.Repositories;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reactive.Linq;
using System.Reactive.Subjects;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace CinemaTickets.UserControls.Halls
{
    /// <summary>
    /// Interaction logic for HallsUserControl.xaml
    /// </summary>
    public partial class HallsUserControl : UserControl, IDisposable
    {
```

```
private readonly Subject<object?> _destroySubject = new();
private readonly IHallRepository _hallRepository;
private bool disposedValue;

public HallsUserControl(IHallRepository hallRepository)
{
    InitializeComponent();
    _hallRepository = hallRepository;
    HallReactiveUtils.CancelEditHallObservable.TakeUntil(_destroySubject).Subscribe(control =>
    {
        ShowHalls();
        AddButton.Visibility = Visibility.Visible;
    });

    HallReactiveUtils.EditHallObservable.TakeUntil(_destroySubject).Subscribe(control =>
    {
        foreach (SingleHallUserControl hallControl in HallsWrapPanel.Children)
        {
            if (hallControl == control)
            {
                continue;
            }

            hallControl.HideButtons();
        }
        AddButton.Visibility = Visibility.Hidden;
    });

    HallReactiveUtils.SaveHallObservable.TakeUntil(_destroySubject).Subscribe(async hall =>
    {
        if (hall.Id is not null)
        {
            Hall hallDb = _hallRepository.GetHallById(hall.Id)!;
            if (hallDb.RoomNumber != hall.RoomNumber)
            {
                if (_hallRepository.ExistsByRoomNumber(hall.RoomNumber))
                {
                    MessageBox.Show("Hall already exists");
                    return;
                }
            }
            await _hallRepository.UpdateHallAsync(hall);
            MessageBox.Show("Hall has been updated successfully");
        }
        else
        {
            if (_hallRepository.ExistsByRoomNumber(hall.RoomNumber))
            {
                MessageBox.Show("Hall already exists");
                return;
            }
            await _hallRepository.AddHallAsync(hall);
            MessageBox.Show("Hall has been added successfully");
        }
```

```
        ShowHalls();
        AddButton.Visibility = Visibility.Visible;
    });

    ShowHalls();

}
private void ShowHalls()
{
    HallsWrapPanel.Children.Clear();
    List<Models.Hall> halls = _hallRepository.GetHalls();
    var singleHallUserControls = halls.Select(hall =>
        new SingleHallUserControl(hall)
    ).ToList();
    singleHallUserControls.ForEach(hallUserControl => HallsWrapPanel.Children.Add(hallUserContro

}

private void OnAddHallClick(object sender, RoutedEventArgs e)
{
    foreach (SingleHallUserControl item in HallsWrapPanel.Children)
    {
        item.HideButtons();
    }
    HallsWrapPanel.Children.Add(new SingleHallUserControl(null));
    AddButton.Visibility = Visibility.Hidden;
}


protected virtual void Dispose(bool disposing)
{
    if (!disposedValue)
    {
        if (disposing)
        {
            _destroySubject.OnNext(null);
        }

        // TODO: free unmanaged resources (unmanaged objects) and override finalizer
        // TODO: set large fields to null
        disposedValue = true;
    }
}

// // TODO: override finalizer only if 'Dispose(bool disposing)' has code to free unmanaged reso
// ~HallsUserControl()
// {
//     // Do not change this code. Put cleanup code in 'Dispose(bool disposing)' method
//     Dispose(disposing: false);
// }

public void Dispose()
{
    // Do not change this code. Put cleanup code in 'Dispose(bool disposing)' method
```

```
            Dispose(disposing: true);
            GC.SuppressFinalize(this);
        }
    }
}

<UserControl x:Class="CinemaTickets.UserControls.Halls.SingleHallUserControl"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:CinemaTickets.UserControls.Halls"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800">
    <StackPanel Orientation="Vertical">
        <StackPanel Orientation="Horizontal">
            <Label Content="Size" Width="60"/>
            <TextBox x:Name="HallSizeTextBlock" MinWidth="40" Width="100" PreviewTextInput="NumberValida
        </StackPanel>
        <StackPanel Orientation="Horizontal">
            <Label Content="Number" Width="60"/>
            <TextBox x:Name="HallNumberTextBlock" MinWidth="40" Width="100" PreviewTextInput="NumberVali
        </StackPanel>
        <StackPanel Orientation="Horizontal">
            <Button x:Name="EditButton" Content="Edit" Margin="0,10,0,0" Width="160" HorizontalAlignment
            <Button x:Name="CancelButton" Content="Cancel" Margin="0,10,0,0" Width="80" HorizontalAlignm
            <Button x:Name="SaveButton" Content="Save" Margin="0,10,0,0" Width="80" HorizontalAlignment=
        </StackPanel>
    </StackPanel>
</UserControl>

using CinemaTickets.Models;
using CinemaTickets.Reactive;
using CinemaTickets.Utils;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace CinemaTickets.UserControls.Halls
{
    /// <summary>
    /// Interaction logic for SingleHallUserControl.xaml
    /// </summary>
```

```
public partial class SingleHallUserControl : UserControl
{
    private bool _editing = false;
    public Hall Hall { get; set; }
    public SingleHallUserControl(Hall? hall)
    {
        InitializeComponent();
        if (hall is not null)
        {
            Hall = hall;
            LoadContent();
            SetEditControls(false);
        }
        else
        {
            Hall = new()
            {
                Size = 20
            };
            SetEditControls(true);
        }
    }

    private void SetEditControls(bool edit)
    {
        _editing = edit;

        HallNumberTextBlock.IsEnabled = edit;
        HallSizeTextBlock.IsEnabled = edit;

        SaveButton.Visibility = edit ? Visibility.Visible : Visibility.Collapsed;
        CancelButton.Visibility = edit ? Visibility.Visible : Visibility.Collapsed;
        EditButton.Visibility = !edit ? Visibility.Visible : Visibility.Collapsed;

        SaveButton.IsEnabled = _editing && IsValidHall();
    }

    private bool IsValidHall()
    {
        var parsedSize = int.TryParse(HallSizeTextBlock.Text.Trim(), out int size);
        var parsedRoom = int.TryParse(HallNumberTextBlock.Text.Trim(), out int room);

        if (!parsedSize || !parsedRoom)
        {
            return false;
        }
        var tempHall = new Hall()
        {
            Size = size,
            RoomNumber = room
        };
        return tempHall.IsValid();
    }
```

```
        private void LoadContent()
        {
            HallSizeTextBlock.Text = Hall.Size.ToString();
            HallNumberTextBlock.Text = Hall.RoomNumber.ToString();
        }


        private void NumberValidationTextBox(object sender, TextCompositionEventArgs e)
        {
            e.Handled = ValidatorUtils.IsNumber(e.Text);
        }


        private void OnCancelClick(object sender, RoutedEventArgs e)
        {
            HallReactiveUtils.OnCancelEditHall(this);
        }

        private void OnEditClick(object sender, RoutedEventArgs e)
        {
            SetEditControls(true);
            HallReactiveUtils.OnEditHall(this);
        }

        public void HideButtons()
        {
            SaveButton.Visibility = Visibility.Collapsed;
            CancelButton.Visibility = Visibility.Collapsed;
            EditButton.Visibility = Visibility.Collapsed;
        }

        private void OnSaveClick(object sender, RoutedEventArgs e)
        {
            Hall.RoomNumber = int.Parse(HallNumberTextBlock.Text);
            Hall.Size = int.Parse(HallSizeTextBlock.Text);
            HallReactiveUtils.OnSaveHall(Hall);
        }

        private void OnHallSizeTextChanged(object sender, TextChangedEventArgs e)
        {
            SaveButton.IsEnabled = IsValidHall();
        }

        private void OnHallNumberTextChanged(object sender, TextChangedEventArgs e)
        {
            SaveButton.IsEnabled = IsValidHall();
        }

    }
}

<UserControl x:Class="CinemaTickets.UserControls.Home.HomeUserControl"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

```
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:CinemaTickets.UserControls.Home"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800">
    <Grid>
            <TextBlock x:Name="MessageTextBlock" HorizontalAlignment="Center" VerticalAlignment="Center'
    </Grid>
</UserControl>

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace CinemaTickets.UserControls.Home
{
    /// <summary>
    /// Interaction logic for HomeUserControl.xaml
    /// </summary>
    public partial class HomeUserControl : UserControl
    {
        public HomeUserControl(string login)
        {
            InitializeComponent();
            SetMessage(login);
        }

        private void SetMessage(string login)
        {
            MessageTextBlock.Text = $"Welcome {login}";
        }
    }
}

<UserControl x:Class="CinemaTickets.Pages.Login.ClientLoginUserControl"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:CinemaTickets.Pages.Login"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800">
    <Grid>
```

```
        <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
            <StackPanel Orientation="Horizontal" Margin="0,0,0,10">
                <Label Content="Login" Width="60"/>
                <TextBox x:Name="LoginTextBox" Width="150" TextChanged="OnLoginTextChanged" />
            </StackPanel>
            <StackPanel Orientation="Horizontal" Margin="0,0,0,10">
                <Label Content="Password" Width="60"/>
                <PasswordBox x:Name="PasswordBox" Width="150" PasswordChanged="OnPasswordChanged"/>
            </StackPanel>

            <Button x:Name="SignInLoginButton" Content="Sign in" Width="80" Margin="0,0,0,10" Click="OnS
            <Button x:Name="SignUpLoginButton" Content="Sign up" Width="80" Click="OnSignUpClick"/>
        </StackPanel>
    </Grid>
</UserControl>

using CinemaTickets.Authentication;
using CinemaTickets.Models;
using CinemaTickets.Reactive;
using CinemaTickets.Repositories;
using CinemaTickets.Utils;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace CinemaTickets.Pages.Login
{
    /// <summary>
    /// Interaction logic for ClientLoginUserControl.xaml
    /// </summary>
    public partial class ClientLoginUserControl : UserControl
    {
        private readonly IAuthStore _authStore;
        private readonly IAuthService _authService;
        private readonly IClientRepository _clientRepository;
        private readonly IPasswordCryption _passwordCryption;

        public ClientLoginUserControl(IAuthStore authStore, IAuthService authService, IClientRepository
        {
            _authStore = authStore;
            _authService = authService;
            _clientRepository = clientRepository;
            _passwordCryption = passwordCryption;
```

```
        InitializeComponent();
        UpdateIsEnabledSignInButton();
        UpdateIsEnabledSignUpButton();
    }

    private void UpdateIsEnabledSignInButton()
    {
        SignInLoginButton.IsEnabled = ValidatorUtils.IsValidLogin(LoginTextBox.Text) && !string.IsNu
    }

    private void UpdateIsEnabledSignUpButton()
    {
        SignUpLoginButton.IsEnabled = ValidatorUtils.IsValidLogin(LoginTextBox.Text) && ValidatorUti
    }

    private void OnLoginTextChanged(object sender, TextChangedEventArgs e)
    {
        UpdateIsEnabledSignInButton();
        UpdateIsEnabledSignUpButton();
    }


    private void OnPasswordChanged(object sender, RoutedEventArgs e)
    {
        UpdateIsEnabledSignInButton();
        UpdateIsEnabledSignUpButton();
    }

    private void OnSignInClick(object sender, RoutedEventArgs e)
    {
        SignInLoginButton.IsEnabled = false;
        var login = LoginTextBox.Text;
        var password = PasswordBox.Password;
        long? id = _authService.Authenticate(Models.AccountType.CLIENT, login, password);
        if (id is not null)
        {
            _authStore.Store(Models.AccountType.CLIENT, login, (long)id);
            MessageBox.Show("Login succeed");
            LoginReactiveUtils.OnLogin(AccountType.CLIENT);
        }
        else
        {
            MessageBox.Show("Incorrect login or password");
        }
        SignInLoginButton.IsEnabled = true;
    }
    private async void OnSignUpClick(object sender, RoutedEventArgs e)
    {
        SignUpLoginButton.IsEnabled = false;
        var login = LoginTextBox.Text;
        var password = PasswordBox.Password;
        if (_clientRepository.ExistsByLoginIgnoreCase(login))
        {
            MessageBox.Show("Login already in use");
```

```
            }
            else
            {
                var client = new Client()
                {
                    Login = login,
                    Password = _passwordCryption.EncryptPassword(password)
                };
                client = await _clientRepository.AddClientAsync(client);
                MessageBox.Show("Account created. You can sign in.");
            }

            SignUpLoginButton.IsEnabled = true;
        }

    }
}

<UserControl x:Class="CinemaTickets.Pages.Login.EmployeeLoginUserControl"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:CinemaTickets.Pages.Login"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800">
    <Grid>
        <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
            <StackPanel Orientation="Horizontal" Margin="0,0,0,10">
                <Label Content="Login" Width="60"/>
                <TextBox x:Name="LoginTextBox" Width="150" TextChanged="OnLoginTextChanged" />
            </StackPanel>
            <StackPanel Orientation="Horizontal" Margin="0,0,0,10">
                <Label Content="Password" Width="60"/>
                <PasswordBox x:Name="PasswordBox" Width="150" PasswordChanged="OnPasswordChanged"/>
            </StackPanel>

            <Button x:Name="SignInLoginButton" Content="Sign in" Width="80" Margin="0,0,0,10" Click="OnS
        </StackPanel>
    </Grid>
</UserControl>

using CinemaTickets.Authentication;
using CinemaTickets.Models;
using CinemaTickets.Reactive;
using CinemaTickets.Repositories;
using CinemaTickets.Utils;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
```

```
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace CinemaTickets.Pages.Login
{
    /// <summary>
    /// Interaction logic for EmployeeLoginUserControl.xaml
    /// </summary>
    public partial class EmployeeLoginUserControl : UserControl
    {

        private readonly IAuthStore _authStore;
        private readonly IAuthService _authService;
        private readonly IEmployeeRepository _employeeRepository;
        private readonly IPasswordCryption _passwordCryption;

        public EmployeeLoginUserControl(IAuthStore authStore, IAuthService authService, IEmployeeReposit
        {
            _authStore = authStore;
            _authService = authService;
            _employeeRepository = employeeRepository;
            _passwordCryption = passwordCryption;
            InitializeComponent();
            UpdateIsEnabledSignInButton();
        }

        private void UpdateIsEnabledSignInButton()
        {
            SignInLoginButton.IsEnabled = ValidatorUtils.IsValidLogin(LoginTextBox.Text) && !string.IsNu
        }


        private void OnLoginTextChanged(object sender, TextChangedEventArgs e)
        {
            UpdateIsEnabledSignInButton();
        }


        private void OnPasswordChanged(object sender, RoutedEventArgs e)
        {
            UpdateIsEnabledSignInButton();
        }

        private void OnSignInClick(object sender, RoutedEventArgs e)
        {
            SignInLoginButton.IsEnabled = false;
            var login = LoginTextBox.Text;
            var password = PasswordBox.Password;
            long? id = _authService.Authenticate(AccountType.EMPLOYEE, login, password);
```

```
            if (id is not null)
            {
                _authStore.Store(AccountType.EMPLOYEE, login, (long)id);
                MessageBox.Show("Login succeed");
                LoginReactiveUtils.OnLogin(AccountType.CLIENT);
            }
            else
            {
                MessageBox.Show("Incorrect login or password");
            }
            SignInLoginButton.IsEnabled = true;
        }
    }
}

<UserControl x:Class="CinemaTickets.Pages.Login.MainLoginUserControl"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:CinemaTickets.Pages.Login"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition/>
            <RowDefinition Height="9*"/>
        </Grid.RowDefinitions>

        <StackPanel Grid.Row="0" Grid.Column="0" Orientation="Horizontal" Height="20" HorizontalAlignmen
            <Button x:Name="ClientButton" Content="Client" Width="100" Click="OnClientClick"/>
            <Button x:Name="EmployeeButton" Content="Employee" Width="100" Click="OnEmployeeClick"/>
        </StackPanel>

        <ContentControl Grid.Row="1" Grid.Column="0" x:Name="LoginContentControl" />
    </Grid>
</UserControl>

using CinemaTickets.Authentication;
using CinemaTickets.Repositories;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
```

```
namespace CinemaTickets.Pages.Login
{
    /// <summary>
    /// Interaction logic for MainLoginUserControl.xaml
    /// </summary>
    public partial class MainLoginUserControl : UserControl
    {
        enum LoginType
        {
            CLIENT,
            EMPLOYEE
        }

        private LoginType _currentLoginType;
        private readonly IClientRepository _clientRepository;
        private readonly IAuthStore _authStore;
        private readonly IAuthService _authService;
        private readonly IPasswordCryption _passwordCryption;
        private readonly IEmployeeRepository _employeeRepository;

        public MainLoginUserControl(IAuthStore authStore, IAuthService authService, IClientRepository cl
            IEmployeeRepository employeeRepository)
        {
            _clientRepository = clientRepository;
            _authStore = authStore;
            _authService = authService;
            _passwordCryption = passwordCryption;
            _employeeRepository = employeeRepository;
            InitializeComponent();
            InitStartupPage();
        }

        private void InitStartupPage()
        {
            SetLoginPage(LoginType.CLIENT);
        }

        private void SetLoginPage(LoginType loginType)
        {
            _currentLoginType = loginType;
            if (LoginType.CLIENT == loginType)
            {
                LoginContentControl.Content = new ClientLoginUserControl(_authStore, _authService, _clie
                ClientButton.IsEnabled = false;
                EmployeeButton.IsEnabled = true;
            }
            else
            {
                LoginContentControl.Content = new EmployeeLoginUserControl(_authStore, _authService,_emp
                EmployeeButton.IsEnabled = false;
                ClientButton.IsEnabled = true;
            }
        }
```

```
        private void OnClientClick(object sender, RoutedEventArgs e)
        {
            SetLoginPage(LoginType.CLIENT);
        }

        private void OnEmployeeClick(object sender, RoutedEventArgs e)
        {
            SetLoginPage(LoginType.EMPLOYEE);
        }
    }
}

using CinemaTickets.Reactive;
using CinemaTickets.Repositories;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Reactive.Subjects;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Reactive.Linq;
using CinemaTickets.Authentication;
using CinemaTickets.Models;

namespace CinemaTickets.UserControls.Movies
{
    /// <summary>
    /// Interaction logic for MoviesUserControl.xaml
    /// </summary>
    public partial class MoviesUserControl : UserControl, IDisposable
    {
        private readonly Subject<object?> _destroySubject = new();
        private readonly IMovieRepository _movieRepository;
        private readonly IAuthStore _authStore;
        private bool disposedValue;

        public MoviesUserControl(IMovieRepository movieRepository, IAuthStore authStore)
        {
            InitializeComponent();
            _movieRepository = movieRepository;
            _authStore = authStore;
            if (_authStore.Type == AccountType.EMPLOYEE)
            {
```

```
        AddButton.Visibility = Visibility.Visible;
        MovieReactiveUtils.DeleteMovieObservable.TakeUntil(_destroySubject).Subscribe(async id =
        {

            if (_movieRepository.IsMovieUsed(id))
            {
                MessageBox.Show("Movie is used");
                return;
            }
            Movie movie = new()
            {
                Id = id
            };
            await _movieRepository.RemoveMovieAsync(movie);
            MessageBox.Show("Deleted successfully");
            ShowMovies();
            AddButton.Visibility = Visibility.Visible;
        });
        MovieReactiveUtils.CancelEditMovieObservable.TakeUntil(_destroySubject).Subscribe(contro
        {
            ShowMovies();
            AddButton.Visibility = Visibility.Visible;
        });
        MovieReactiveUtils.EditMovieObservable.TakeUntil(_destroySubject).Subscribe(control =>
        {
            foreach (SingleMovieUserControl movieControl in MoviesWrapPanel.Children)
            {
                if (movieControl == control)
                {
                    continue;
                }

                movieControl.HideButtons();
            }
            AddButton.Visibility = Visibility.Hidden;
        });
        MovieReactiveUtils.SaveMovieObservable.TakeUntil(_destroySubject).Subscribe(async movie
        {
            if (movie.Id is not null)
            {
                await _movieRepository.UpdateMovieAsync(movie);
                MessageBox.Show("Movie has been updated successfully");
            }
            else
            {
                await _movieRepository.AddMovieAsync(movie);
                MessageBox.Show("Movie has been added successfully");
            }
            ShowMovies();
            AddButton.Visibility = Visibility.Visible;
        });
    }
    else if (_authStore.Type == AccountType.CLIENT)
    {
```

```
        AddButton.Visibility = Visibility.Hidden;
    }
    ShowMovies();
}

private void ShowMovies()
{
    MoviesWrapPanel.Children.Clear();
    List<Models.Movie> movies = _movieRepository.GetMovies();
    var singleMovieUserControls = movies.Select(movie =>
        new SingleMovieUserControl(movie, _authStore.Type != AccountType.EMPLOYEE)
    ).ToList();
    singleMovieUserControls.ForEach(movieUserControl => MoviesWrapPanel.Children.Add(movieUserCo

}

private void OnAddMovieClick(object sender, RoutedEventArgs e)
{
    foreach (SingleMovieUserControl item in MoviesWrapPanel.Children)
    {
        item.HideButtons();
    }
    MoviesWrapPanel.Children.Add(new SingleMovieUserControl(null, _authStore.Type != AccountType
    AddButton.Visibility = Visibility.Hidden;
}

protected virtual void Dispose(bool disposing)
{
    if (!disposedValue)
    {
        if (disposing)
        {
            // TODO: dispose managed state (managed objects)
            _destroySubject.OnNext(null);
        }

        // TODO: free unmanaged resources (unmanaged objects) and override finalizer
        // TODO: set large fields to null
        disposedValue = true;
    }
}

// // TODO: override finalizer only if 'Dispose(bool disposing)' has code to free unmanaged reso
// ~MoviesUserControl()
// {
//     // Do not change this code. Put cleanup code in 'Dispose(bool disposing)' method
//     Dispose(disposing: false);
// }

public void Dispose()
{
    // Do not change this code. Put cleanup code in 'Dispose(bool disposing)' method
    Dispose(disposing: true);
    GC.SuppressFinalize(this);
```

```
        }
    }
}

using CinemaTickets.Models;
using CinemaTickets.Repositories;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Text.RegularExpressions;
using CinemaTickets.Utils;
using CinemaTickets.Reactive;

namespace CinemaTickets.UserControls.Movies
{
    /// <summary>
    /// Interaction logic for SingleMovieUserControl.xaml
    /// </summary>
    public partial class SingleMovieUserControl : UserControl
    {
        private bool _editing = false;
        public Movie Movie { get; set; }
        public SingleMovieUserControl(Movie? movie, bool readonlyView)
        {
            InitializeComponent();
            if (movie is not null)
            {
                Movie = movie;
                LoadContent();
                if (!readonlyView)
                {
                    SetEditControls(false);
                }
            }
            else
            {
                Movie = new()
                {
                    Year = DateTime.Now
                };
                if (!readonlyView)
                {
                    SetEditControls(true);
```

```
        }
    }
    if (readonlyView)
    {
        SetEditControls(false);
        HideButtons();
    }
}

private void SetEditControls(bool edit)
{
    _editing = edit;

    MovieTitleTextBlock.IsEnabled = edit;
    MovieYearDatePicker.IsEnabled = edit;
    MovieDurationTextBlock.IsEnabled = edit;

    SaveButton.Visibility = edit ? Visibility.Visible : Visibility.Collapsed;
    CancelButton.Visibility = edit ? Visibility.Visible : Visibility.Collapsed;
    EditButton.Visibility = !edit ? Visibility.Visible : Visibility.Collapsed;
    DeleteButton.Visibility = !edit ? Visibility.Visible : Visibility.Collapsed;

    SaveButton.IsEnabled = _editing && IsValidMovie();
}

private bool IsValidMovie()
{
    var tempMovie = new Movie()
    {
        Title = MovieTitleTextBlock.Text.Trim(),
        Year = MovieYearDatePicker.SelectedDate == null ? new DateTime() : MovieYearDatePicker.S
        Duration = string.IsNullOrEmpty(MovieDurationTextBlock.Text) ? 0 : int.Parse(MovieDurati
    };
    return tempMovie.IsValid();
}

private void LoadContent()
{
    MovieTitleTextBlock.Text = Movie.Title;
    MovieYearDatePicker.SelectedDate = Movie.Year.Date;
    MovieDurationTextBlock.Text = Movie.Duration.ToString();
}


private void NumberValidationTextBox(object sender, TextCompositionEventArgs e)
{
    e.Handled = ValidatorUtils.IsNumber(e.Text);
}

private void OnDeleteClick(object sender, RoutedEventArgs e)
{
    MovieReactiveUtils.OnDeleteMovie(Movie.Id!.Value);
}
```

```
        private void OnCancelClick(object sender, RoutedEventArgs e)
        {
            MovieReactiveUtils.OnCancelEditMovie(this);
        }

        private void OnEditClick(object sender, RoutedEventArgs e)
        {
            SetEditControls(true);
            MovieReactiveUtils.OnEditMovie(this);
        }

        public void HideButtons()
        {
            SaveButton.Visibility = Visibility.Collapsed;
            CancelButton.Visibility = Visibility.Collapsed;
            EditButton.Visibility = Visibility.Collapsed;
            DeleteButton.Visibility = Visibility.Collapsed;
        }

        private void OnSaveClick(object sender, RoutedEventArgs e)
        {
            Movie.Title = MovieTitleTextBlock.Text.Trim();
            Movie.Year = MovieYearDatePicker.SelectedDate!.Value.Date;
            Movie.Duration = int.Parse(MovieDurationTextBlock.Text);
            MovieReactiveUtils.OnSaveMovie(Movie);
        }

        private void OnMovieTitleTextChanged(object sender, TextChangedEventArgs e)
        {
            SaveButton.IsEnabled = IsValidMovie();
        }

        private void OnMovieDurationTextChanged(object sender, TextChangedEventArgs e)
        {
            SaveButton.IsEnabled = IsValidMovie();
        }

        private void OnMovieYearSelectedDateChanged(object sender, SelectionChangedEventArgs e)
        {
            SaveButton.IsEnabled = IsValidMovie();
        }
    }
}

<UserControl x:Class="CinemaTickets.UserControls.Movies.SingleMovieUserControl"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:CinemaTickets.UserControls.Movies"
             mc:Ignorable="d" d:DesignWidth="160" Height="110">
    <StackPanel Orientation="Vertical">
        <StackPanel Orientation="Horizontal">
            <Label Content="Title" Width="60"/>
```

```
            <TextBox x:Name="MovieTitleTextBlock" MinWidth="40" Width="100" TextChanged="OnMovieTitleTex
        </StackPanel>
        <StackPanel Orientation="Horizontal">
            <Label Content="Date"  Width="60"/>
            <!--<TextBox x:Name="MovieYearTextBlock" MinWidth="40" Width="100"/>-->
            <DatePicker x:Name="MovieYearDatePicker" MinWidth="40" Width="100" SelectedDateChanged="OnMo
        </StackPanel>
        <StackPanel Orientation="Horizontal">
            <Label Content="Duration" Width="60"/>
            <TextBox x:Name="MovieDurationTextBlock" MinWidth="40" Width="100" PreviewTextInput="NumberV
        </StackPanel>
        <StackPanel Orientation="Horizontal">
            <Button x:Name="DeleteButton" Content="Delete" Margin="0,10,0,0" Width="80" HorizontalAlignm
            <Button x:Name="EditButton" Content="Edit" Margin="0,10,0,0" Width="80" HorizontalAlignment=
            <Button x:Name="CancelButton" Content="Cancel" Margin="0,10,0,0" Width="80" HorizontalAlignm
            <Button x:Name="SaveButton" Content="Save" Margin="0,10,0,0" Width="80" HorizontalAlignment=
        </StackPanel>
    </StackPanel>
</UserControl>

<UserControl x:Class="CinemaTickets.UserControls.Movies.MoviesUserControl"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:CinemaTickets.UserControls.Movies"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="9*"></RowDefinition>
            <RowDefinition></RowDefinition>
        </Grid.RowDefinitions>
        <WrapPanel Grid.Row="0" x:Name="MoviesWrapPanel"></WrapPanel>
        <Button x:Name="AddButton" Grid.Row="1" Content="Add" Click="OnAddMovieClick"/>
    </Grid>
</UserControl>

using CinemaTickets.Models;
using CinemaTickets.Reactive;
using CinemaTickets.Repositories;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reactive.Subjects;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
```

```
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Reactive.Linq;
using CinemaTickets.Authentication;
using CinemaTickets.Exceptions;

namespace CinemaTickets.UserControls.Seances
{
    /// <summary>
    /// Interaction logic for SeancesUserControl.xaml
    /// </summary>
    public partial class SeancesUserControl : UserControl, IDisposable
    {
        private readonly IMovieRepository _movieRepository;
        private readonly IHallRepository _hallRepository;
        private readonly ISeanceRepository _seanceRepository;
        private readonly IAuthStore _authStore;
        private readonly ISeanceRegistrationRepository? _seanceRegistrationRepository;
        private readonly Subject<object?> _destroySubject = new();
        private bool disposedValue;

        public SeancesUserControl(IMovieRepository movieRepository, IHallRepository hallRepository, ISea
            IAuthStore authStore,
            ISeanceRegistrationRepository? seanceRegistrationRepository
            )
        {
            InitializeComponent();
            _movieRepository = movieRepository;
            _hallRepository = hallRepository;
            _seanceRepository = seanceRepository;
            _authStore = authStore;
            _seanceRegistrationRepository = seanceRegistrationRepository;

            if (_authStore.Type == AccountType.EMPLOYEE)
            {
                AddButton.Visibility = Visibility.Visible;
                SeanceReactiveUtils.CancelEditSeanceObservable.TakeUntil(_destroySubject).Subscribe(cont
                {
                    ShowSeances();
                    AddButton.Visibility = Visibility.Visible;
                });

                SeanceReactiveUtils.SaveSeanceObservable.TakeUntil(_destroySubject).Subscribe(async sear
                {
                    try
                    {
                        await _seanceRepository.AddSeanceAsync(seance);
                    }
                    catch (Microsoft.EntityFrameworkCore.DbUpdateException ex)
                    {
                        if (ex.InnerException is Microsoft.Data.SqlClient.SqlException && ex.InnerExcept
                        {
                            _seanceRepository.Detach(seance);
                            MessageBox.Show("Seance already exists");
```

```
                    return;
                }
            }
            MessageBox.Show("Seance has been added successfully");
            ShowSeances();
            AddButton.Visibility = Visibility.Visible;
        }
        );
    }
    else
    {
        AddButton.Visibility = Visibility.Collapsed;
        if (_seanceRegistrationRepository is null)
        {
            throw new ArgumentNullException(nameof(seanceRegistrationRepository));
        }

        SeanceReactiveUtils.AttendSeanceObservable.TakeUntil(_destroySubject).Subscribe(async se
        {
            if (_authStore.Id is null)
            {
                throw new NotLoggedException();
            }
            ClientsMoviesHall clientsMoviesHall = new()
            {
                IdClient = (long)_authStore.Id,
                IdHall = seance.IdHall,
                IdMovie = seance.IdMovie,
            };
            await _seanceRegistrationRepository!.AttendAsync(clientsMoviesHall);
            MessageBox.Show("Attended");
            ShowSeances();
        });

        SeanceReactiveUtils.CancelAttendSeanceObservable.TakeUntil(_destroySubject).Subscribe(as
        {
            if (_authStore.Id is null)
            {
                throw new NotLoggedException();
            }
            ClientsMoviesHall clientsMoviesHall = new()
            {
                IdClient = (long)_authStore.Id,
                IdHall = seance.IdHall,
                IdMovie = seance.IdMovie,
            };
            await _seanceRegistrationRepository!.RemoveAttendAsync(clientsMoviesHall);
            MessageBox.Show("Attend canceled");
            ShowSeances();
        });
    }
    ShowSeances();
}
```

```
private void ShowSeances()
{
    SeancesWrapPanel.Children.Clear();
    List<MoviesHall> seances = _seanceRepository.GetSeances();

    if (_authStore.Type == AccountType.EMPLOYEE)
    {
        foreach (var seance in seances)
        {
            SeancesWrapPanel.Children.Add(new SingleSeanceUserControl(seance));
        }
    }
    else
    {
        if (_authStore.Id is null)
        {
            throw new NotLoggedException();
        }

        var clientSeances = _seanceRegistrationRepository!.GetSeances((long)_authStore.Id);
        foreach (var seance in seances)
        {
            var isAttending = IsAttending(clientSeances, seance.IdMovie, seance.IdHall);
            SeancesWrapPanel.Children.Add(new SingleSeanceUserControl(seance, isAttending));
        }
    }
}

private bool IsAttending(List<ClientsMoviesHall> clientSeances, long idMoie, long idHall)
{
    return clientSeances.Any(clientSeance => clientSeance.IdMovie == idMoie && clientSeance.IdHa
}

private void OnAddSeanceClick(object sender, RoutedEventArgs e)
{
    var movies = _movieRepository.GetMovies();
    var halls = _hallRepository.GetHalls();
    SeancesWrapPanel.Children.Add(new NewSeanceUserControl(movies, halls));
    AddButton.Visibility = Visibility.Hidden;
}

protected virtual void Dispose(bool disposing)
{
    if (!disposedValue)
    {
        if (disposing)
        {
            // TODO: dispose managed state (managed objects)
            _destroySubject.OnNext(null);
        }

        // TODO: free unmanaged resources (unmanaged objects) and override finalizer
        // TODO: set large fields to null
        disposedValue = true;
```

```
            }
        }

        // // TODO: override finalizer only if 'Dispose(bool disposing)' has code to free unmanaged reso
        // ~SeancesUserControl()
        // {
        //     // Do not change this code. Put cleanup code in 'Dispose(bool disposing)' method
        //     Dispose(disposing: false);
        // }

        public void Dispose()
        {
            // Do not change this code. Put cleanup code in 'Dispose(bool disposing)' method
            Dispose(disposing: true);
            GC.SuppressFinalize(this);
        }
    }
}

<UserControl x:Class="CinemaTickets.UserControls.Seances.SingleSeanceUserControl"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:CinemaTickets.UserControls.Seances"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800">
    <StackPanel Orientation="Vertical">
        <StackPanel Orientation="Horizontal">
            <TextBlock Text="Movie" Width="100" FontWeight="Bold"/>
            <TextBlock x:Name="MovieTitleTextBlock" Width="200" FontWeight="Bold"/>
        </StackPanel>
        <StackPanel Orientation="Horizontal">
            <TextBlock Text="Room number" Width="100" FontWeight="Bold"/>
            <TextBlock x:Name="HallRoomTextBlock" Width="200"/>
        </StackPanel>
        <StackPanel Orientation="Horizontal">
            <TextBlock Text="Room size" Width="100" FontWeight="Bold"/>
            <TextBlock x:Name="HallSizeTextBlock" Width="200"/>
        </StackPanel>
        <StackPanel Orientation="Horizontal">
            <TextBlock Text="Start date" Width="100" FontWeight="Bold"/>
            <TextBlock x:Name="StartDateTextBlock" Width="200"/>
        </StackPanel>
        <StackPanel Orientation="Horizontal">
            <TextBlock Text="End date" Width="100" FontWeight="Bold"/>
            <TextBlock x:Name="EndDateTextBlock" Width="200"/>
        </StackPanel>
        <Button x:Name="AttendButton" Click="AttendButtonClick"/>
    </StackPanel>
</UserControl>

using CinemaTickets.Models;
using CinemaTickets.Reactive;
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace CinemaTickets.UserControls.Seances
{
    /// <summary>
    /// Interaction logic for SingleSeanceUserControl.xaml
    /// </summary>
    public partial class SingleSeanceUserControl : UserControl
    {
        private readonly MoviesHall _seance;
        private bool? _attending;
        private Movie Movie { get { return _seance.IdMovieNavigation; } }
        private Hall Hall { get { return _seance.IdHallNavigation; } }

        public SingleSeanceUserControl(MoviesHall seance, bool? attending = null)
        {
            InitializeComponent();
            _seance = seance;
            _attending = attending;
            LoadContent();
            ShowButtons();
        }

        private void ShowButtons()
        {
            if (_attending is null)
            {
                AttendButton.Visibility = Visibility.Collapsed;
                return;
            }

            AttendButton.Visibility = Visibility.Visible;
            AttendButton.Content = (bool)_attending ? "Cancel attend" : "Attend";
        }

        private void LoadContent()
        {
            MovieTitleTextBlock.Text = Movie.ToString();
            HallRoomTextBlock.Text = Hall.RoomNumber.ToString();
            HallSizeTextBlock.Text = Hall.Size.ToString();
            StartDateTextBlock.Text = _seance.StartTime.ToString();
```

```
            EndDateTextBlock.Text = _seance.EndTime.ToString();
        }

        private void AttendButtonClick(object sender, RoutedEventArgs e)
        {

            if (_attending is null)
            {
                return;
            }

            if ((bool)_attending)
            {
                _attending = false;
                SeanceReactiveUtils.OnCancelAttendSeance(_seance);

            }
            else
            {
                _attending = true;
                SeanceReactiveUtils.OnAttendSeance(_seance);
            }
            ShowButtons();
        }
    }
}

<UserControl x:Class="CinemaTickets.UserControls.Seances.NewSeanceUserControl"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:CinemaTickets.UserControls.Seances"
             xmlns:xctk="http://schemas.xceed.com/wpf/xaml/toolkit"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800">
    <StackPanel Orientation="Vertical">
        <StackPanel Orientation="Horizontal">
            <Label Content="Movie" Width="60"/>
            <ComboBox x:Name="MoviesComboBox" Width="220" SelectionChanged="MoviesComboBoxSelectionChang
            </ComboBox>
        </StackPanel>
        <StackPanel Orientation="Horizontal">
            <Label Content="Hall"  Width="60"/>
            <ComboBox x:Name="HallsComboBox" Width="220" SelectionChanged="HallsComboBoxSelectionChanged
            </ComboBox>
        </StackPanel>
        <StackPanel Orientation="Horizontal">
            <Label Content="Date" Width="60"/>
            <xctk:DateTimePicker x:Name="SeanceDateDatePicker" MinWidth="40" Width="220" ValueChanged="S
        </StackPanel>
        <StackPanel Orientation="Horizontal">
            <Button x:Name="CancelButton" Content="Cancel" Margin="0,10,0,0" Width="140" HorizontalAligr
            <Button x:Name="SaveButton" Content="Save" Margin="0,10,0,0" Width="140" HorizontalAlignment
```

```
        </StackPanel>
    </StackPanel>
</UserControl>

using CinemaTickets.Exceptions;
using CinemaTickets.Models;
using CinemaTickets.Reactive;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace CinemaTickets.UserControls.Seances
{
    /// <summary>
    /// Interaction logic for NewSeanceUserControl.xaml
    /// </summary>
    public partial class NewSeanceUserControl : UserControl
    {
        private readonly List<Movie> _movies;
        private readonly List<Hall> _halls;

        public NewSeanceUserControl(List<Movie> movies, List<Hall> halls)
        {
            InitializeComponent();
            _movies = movies;
            _halls = halls;
            MoviesComboBox.ItemsSource = _movies;
            HallsComboBox.ItemsSource = _halls;
            UpdateSaveButtonEnabled();
        }

        private void OnCancelClick(object sender, RoutedEventArgs e)
        {
            SeanceReactiveUtils.OnCancelEditSeance(this);
        }

        private void OnSaveClick(object sender, RoutedEventArgs e)
        {
            Movie? movie = (MoviesComboBox.SelectedItem as Movie);
            Hall? hall = (HallsComboBox.SelectedItem as Hall);
            DateTime? startTime = SeanceDateDatePicker.Value;
            if (movie is null || hall is null || startTime is null || movie.Id is null || hall.Id is nul
            {
```

```
        throw new UnexpectedValidationException();
    }
    else
    {
        MoviesHall seance = new()
        {
            StartTime = (DateTime)startTime,
            EndTime = ((DateTime)startTime).AddMinutes(movie.Duration),
            IdMovie = (long)movie.Id,
            IdHall = (long)hall.Id,
        };
        SeanceReactiveUtils.OnSaveSeance(seance);
    }
}

private void OnSeanceDateSelectedDateChanged(object sender, SelectionChangedEventArgs e)
{
    UpdateSaveButtonEnabled();
}

private void UpdateSaveButtonEnabled()
{
    if (MoviesComboBox.SelectedIndex == -1)
    {
        SaveButton.IsEnabled = false;
        return;
    }

    if (HallsComboBox.SelectedIndex == -1)
    {
        SaveButton.IsEnabled = false;
        return;
    }

    if (SeanceDateDatePicker.Value == null || SeanceDateDatePicker.Value <= DateTime.Now)
    {
        SaveButton.IsEnabled = false;
        return;
    }

    SaveButton.IsEnabled = true;
}

private void HallsComboBoxSelectionChanged(object sender, SelectionChangedEventArgs e)
{
    UpdateSaveButtonEnabled();
}

private void MoviesComboBoxSelectionChanged(object sender, SelectionChangedEventArgs e)
{
    UpdateSaveButtonEnabled();
}

private void SeanceDateDatePickerValueChanged(object sender, RoutedPropertyChangedEventArgs<obje
```

```
        {
            UpdateSaveButtonEnabled();
        }
    }
}

<UserControl x:Class="CinemaTickets.UserControls.Seances.SeancesUserControl"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:CinemaTickets.UserControls.Seances"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="9*"></RowDefinition>
            <RowDefinition></RowDefinition>
        </Grid.RowDefinitions>
        <WrapPanel Grid.Row="0" x:Name="SeancesWrapPanel"></WrapPanel>
        <Button x:Name="AddButton" Grid.Row="1" Content="Add" Click="OnAddSeanceClick"/>
    </Grid>
</UserControl>

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;

namespace CinemaTickets.Utils
{
    public class ValidatorUtils
    {
        private ValidatorUtils()
        {

        }

        public static bool IsValidPassword(string password)
        {
            if (string.IsNullOrWhiteSpace(password))
            {
                return false;
            }

            if (password.Contains(' '))
            {
                return false;
            }

            if (password.Length < 7 || password.Length > 254)
            {
```

```
                return false;
            }

            return true;
        }

        public static bool IsValidLogin(string login)
        {
            if (string.IsNullOrWhiteSpace(login))
            {
                return false;
            }

            if (login.Contains(' '))
            {
                return false;
            }

            if (login.Length < 4 || login.Length > 18)
            {
                return false;
            }

            return true;
        }

        public static bool IsNumber(string text)
        {
            Regex regex = new("[^0-9]+");
            return regex.IsMatch(text);
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CinemaTickets
{
    public class CaesarCipher
    {
        private readonly static char[] alphabet = new char[] { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h',
             'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R',

        private CaesarCipher()
        {

        }

        private static char Cipher(char ch, int key)
        {
```

```
            if (!char.IsLetter(ch))
            {

                return ch;
            }

            char d = char.IsUpper(ch) ? 'A' : 'a';
            return (char)((((ch + key) - d) % 26) + d);
        }
        public static string Encrypt(char[] secretMessage, int key)
        {
            string output = string.Empty;

            foreach (char ch in secretMessage) { output += Cipher(ch, key); }

            return output;
        }

        public static string Decrypt(char[] secretMessage, int key)
        {
            return Encrypt(secretMessage, 26 - key);
        }

    }
}

<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>WinExe</OutputType>
    <TargetFramework>net6.0-windows</TargetFramework>
    <Nullable>enable</Nullable>
    <UseWPF>true</UseWPF>
    <StartupObject>CinemaTickets.App</StartupObject>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Extended.Wpf.Toolkit" Version="4.1.0" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="5.0.11" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="5.0.11">
      <PrivateAssets>all</PrivateAssets>
      <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
    </PackageReference>
    <PackageReference Include="Microsoft.Extensions.DependencyInjection" Version="5.0.2" />
    <PackageReference Include="Microsoft.NET.Test.Sdk" Version="17.0.0" />
    <PackageReference Include="MSTest.TestAdapter" Version="2.2.7" />
    <PackageReference Include="MSTest.TestFramework" Version="2.2.7" />
    <PackageReference Include="System.Reactive" Version="5.0.0" />
  </ItemGroup>

</Project>

using CinemaTickets.Authentication;
using CinemaTickets.Models;
```

```
using CinemaTickets.Pages.Login;
using CinemaTickets.Reactive;
using CinemaTickets.Repositories;
using System.Windows;
using System;
using CinemaTickets.UserControls.Home;
using CinemaTickets.Exceptions;
using System.Windows.Controls;
using CinemaTickets.UserControls.Movies;
using System.Reactive.Linq;
using CinemaTickets.UserControls.Employees;
using CinemaTickets.UserControls.Halls;
using CinemaTickets.UserControls.Seances;

namespace CinemaTickets
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        private readonly IAuthStore _authStore;
        private readonly IAuthService _authService;
        private readonly IClientRepository _clientRepository;
        private readonly IEmployeeRepository _employeeRepository;
        private readonly IPasswordCryption _passwordCryption;
        private readonly IMovieRepository _movieRepository;
        private readonly IHallRepository _hallRepository;
        private readonly ISeanceRepository _seanceRepository;
        private readonly ISeanceRegistrationRepository _seanceRegistrationRepository;

        public MainWindow(IAuthStore authStore, IAuthService authService, IClientRepository clientRepos
            ISeanceRegistrationRepository seanceRegistrationRepository)
        {
            _authStore = authStore;
            _authService = authService;
            _clientRepository = clientRepository;
            _passwordCryption = passwordCryption;
            _employeeRepository = employeeRepository;
            _movieRepository = movieRepository;
            _hallRepository = hallRepository;
            _seanceRepository = seanceRepository;
            _seanceRegistrationRepository = seanceRegistrationRepository;
            InitializeComponent();
            InitContentControl();
        }

        private void InitContentControl()
        {
            DisposeCurrentMainContentControl();
            MainContentControl.Content = new MainLoginUserControl(_authStore, _authService, _clientRepos
            LoginReactiveUtils.LoginObservable
                .Take(1)
                .Subscribe(_ =>
```

```
        {
            ShowHome();
            RebuildNavigation();
        });
    }

    private void DisposeCurrentMainContentControl()
    {
        if (MainContentControl.Content is null)
        {
            return;
        }

        if (MainContentControl.Content is IDisposable disposableContent)
        {
            disposableContent.Dispose();
        }
    }

    private void RebuildNavigation()
    {
        ClearNavigation();
        var isLogged = _authStore.Login != null;
        if (isLogged)
        {
            BuildNavigation();
        }
    }

    private void BuildNavigation()
    {
        var hasType = _authStore.Type != null;
        if (!hasType)
        {
            throw new NotLoggedException();
        }
        var type = _authStore.Type == AccountType.CLIENT ? "Client" : "Employee";

        LogoutButton.Content = $"{type} {_authStore.Login}";
        LogoutButton.Visibility = Visibility.Visible;

        var HomeButton = new Button
        {
            Content = "Home",
            MinWidth = 120
        };
        HomeButton.Click += OnHomeClick;
        NavigationStackPanel.Children.Add(HomeButton);
        if (_authStore.Type == AccountType.EMPLOYEE || _authStore.Type == AccountType.CLIENT)
        {
            var MoviesButton = new Button
            {
                Content = "Movies",
                MinWidth = 120
```

```
        };
        MoviesButton.Click += OnMoviesClick;

        var SeancesButton = new Button
        {
            Content = "Seances",
            MinWidth = 120
        };
        SeancesButton.Click += OnSeancesClick;


        NavigationStackPanel.Children.Add(MoviesButton);
        NavigationStackPanel.Children.Add(SeancesButton);
    }
    if (_authStore.Type == AccountType.EMPLOYEE)
    {
        var EmployeesButton = new Button
        {
            Content = "Employees",
            MinWidth = 120
        };
        EmployeesButton.Click += OnEmployeesClick;
        var HallsButton = new Button
        {
            Content = "Halls",
            MinWidth = 120
        };
        HallsButton.Click += OnHallsClick;

        NavigationStackPanel.Children.Add(EmployeesButton);
        NavigationStackPanel.Children.Add(HallsButton);
    }
}

private void ClearNavigation()
{
    LogoutButton.Content = "";
    LogoutButton.Visibility = Visibility.Hidden;
    foreach (var child in NavigationStackPanel.Children)
    {
        if (child is Button button)
        {
            button.Click -= OnHomeClick;
            button.Click -= OnMoviesClick;
            button.Click -= OnEmployeesClick;
            button.Click -= OnHallsClick;
            button.Click -= OnSeancesClick;
        }
    }
    NavigationStackPanel.Children.Clear();
}

private void ShowEmployees()
{
```

```
        if (_authStore.Login == null || _authStore.Type != AccountType.EMPLOYEE)
        {
            throw new NotLoggedException();
        }
        DisposeCurrentMainContentControl();
        MainContentControl.Content = new EmployeesUserControl(_employeeRepository, _passwordCryption
    }

    private void ShowHalls()
    {
        if (_authStore.Login == null || _authStore.Type != AccountType.EMPLOYEE)
        {
            throw new NotLoggedException();
        }
        DisposeCurrentMainContentControl();
        MainContentControl.Content = new HallsUserControl(_hallRepository);
    }

    private void ShowSeances()
    {
        if (_authStore.Login == null)
        {
            throw new NotLoggedException();
        }
        DisposeCurrentMainContentControl();
        MainContentControl.Content = new SeancesUserControl(_movieRepository, _hallRepository, _sear
    }

    private void ShowMovies()
    {
        if (_authStore.Login == null)
        {
            throw new NotLoggedException();
        }
        DisposeCurrentMainContentControl();
        MainContentControl.Content = new MoviesUserControl(_movieRepository, _authStore);
    }

    private void ShowHome()
    {
        if (_authStore.Login == null)
        {
            throw new NotLoggedException();
        }
        DisposeCurrentMainContentControl();
        MainContentControl.Content = new HomeUserControl(_authStore.Login);
    }

    private void OnSeancesClick(object sender, RoutedEventArgs e)
    {
        ShowSeances();
    }

    private void OnHallsClick(object sender, RoutedEventArgs e)
```

```
        {
            ShowHalls();
        }

        private void OnEmployeesClick(object sender, RoutedEventArgs e)
        {
            ShowEmployees();
        }

        private void OnMoviesClick(object sender, RoutedEventArgs e)
        {
            ShowMovies();
        }

        private void OnHomeClick(object sender, RoutedEventArgs e)
        {
            ShowHome();
        }

        private void OnLogoutClick(object sender, RoutedEventArgs e)
        {
            MessageBoxResult messageBoxResult = MessageBox.Show("Do you want to log out?", "Logout", Mes
            switch (messageBoxResult)
            {
                case MessageBoxResult.Yes:
                    _authStore.Logout();
                    RebuildNavigation();
                    InitContentControl();
                    break;
                default: break;
            }
        }
    }
}


Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio Version 17
VisualStudioVersion = 17.0.31825.309
MinimumVisualStudioVersion = 10.0.40219.1
Project("{FAE04EC0-301F-11D3-BF4B-00C04F79EFBC}") = "CinemaTickets", "CinemaTickets.csproj", "{879B0BBE-
EndProject
Global
GlobalSection(SolutionConfigurationPlatforms) = preSolution
Debug|Any CPU = Debug|Any CPU
Release|Any CPU = Release|Any CPU
EndGlobalSection
GlobalSection(ProjectConfigurationPlatforms) = postSolution
{879B0BBE-DC2D-42D8-961D-65338E13F5A4}.Debug|Any CPU.ActiveCfg = Debug|Any CPU
{879B0BBE-DC2D-42D8-961D-65338E13F5A4}.Debug|Any CPU.Build.0 = Debug|Any CPU
{879B0BBE-DC2D-42D8-961D-65338E13F5A4}.Release|Any CPU.ActiveCfg = Release|Any CPU
{879B0BBE-DC2D-42D8-961D-65338E13F5A4}.Release|Any CPU.Build.0 = Release|Any CPU
EndGlobalSection
GlobalSection(SolutionProperties) = preSolution
```

```
HideSolutionNode = FALSE
EndGlobalSection
GlobalSection(ExtensibilityGlobals) = postSolution
SolutionGuid = {6EDB3EFC-8AD6-4357-B641-D22B6FCA7232}
EndGlobalSection
EndGlobal

<Application x:Class="CinemaTickets.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:local="clr-namespace:CinemaTickets"
             Startup="OnStartup">
    <Application.Resources>

    </Application.Resources>
</Application>

<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="Current" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <PropertyGroup />
  <ItemGroup>
    <ApplicationDefinition Update="App.xaml">
      <SubType>Designer</SubType>
    </ApplicationDefinition>
  </ItemGroup>
  <ItemGroup>
    <Compile Update="UserControls\Employees\EmployeesUserControl.xaml.cs">
      <SubType>Code</SubType>
    </Compile>
    <Compile Update="UserControls\Employees\SingleEmployeeUserControl.xaml.cs">
      <SubType>Code</SubType>
    </Compile>
    <Compile Update="UserControls\Halls\HallsUserControl.xaml.cs">
      <SubType>Code</SubType>
    </Compile>
    <Compile Update="UserControls\Halls\SingleHallUserControl.xaml.cs">
      <SubType>Code</SubType>
    </Compile>
    <Compile Update="UserControls\Home\HomeUserControl.xaml.cs">
      <SubType>Code</SubType>
    </Compile>
    <Compile Update="UserControls\Login\ClientLoginUserControl.xaml.cs">
      <SubType>Code</SubType>
    </Compile>
    <Compile Update="UserControls\Login\EmployeeLoginUserControl.xaml.cs">
      <SubType>Code</SubType>
    </Compile>
    <Compile Update="UserControls\Login\MainLoginUserControl.xaml.cs">
      <SubType>Code</SubType>
    </Compile>
    <Compile Update="UserControls\Movies\MoviesUserControl.xaml.cs">
      <SubType>Code</SubType>
    </Compile>
    <Compile Update="UserControls\Movies\SingleMovieUserControl.xaml.cs">
```

```
    <SubType>Code</SubType>
  </Compile>
  <Compile Update="UserControls\Seances\NewSeanceUserControl.xaml.cs">
    <SubType>Code</SubType>
  </Compile>
  <Compile Update="UserControls\Seances\SeancesUserControl.xaml.cs">
    <SubType>Code</SubType>
  </Compile>
  <Compile Update="UserControls\Seances\SingleSeanceUserControl.xaml.cs">
    <SubType>Code</SubType>
  </Compile>
</ItemGroup>
<ItemGroup>
  <Page Update="MainWindow.xaml">
    <SubType>Designer</SubType>
  </Page>
  <Page Update="UserControls\Employees\EmployeesUserControl.xaml">
    <SubType>Designer</SubType>
  </Page>
  <Page Update="UserControls\Employees\SingleEmployeeUserControl.xaml">
    <SubType>Designer</SubType>
  </Page>
  <Page Update="UserControls\Halls\HallsUserControl.xaml">
    <SubType>Designer</SubType>
  </Page>
  <Page Update="UserControls\Halls\SingleHallUserControl.xaml">
    <SubType>Designer</SubType>
  </Page>
  <Page Update="UserControls\Home\HomeUserControl.xaml">
    <SubType>Designer</SubType>
  </Page>
  <Page Update="UserControls\Login\ClientLoginUserControl.xaml">
    <SubType>Designer</SubType>
  </Page>
  <Page Update="UserControls\Login\EmployeeLoginUserControl.xaml">
    <SubType>Designer</SubType>
  </Page>
  <Page Update="UserControls\Login\MainLoginUserControl.xaml">
    <SubType>Designer</SubType>
  </Page>
  <Page Update="UserControls\Movies\MoviesUserControl.xaml">
    <SubType>Designer</SubType>
  </Page>
  <Page Update="UserControls\Movies\SingleMovieUserControl.xaml">
    <SubType>Designer</SubType>
  </Page>
  <Page Update="UserControls\Seances\NewSeanceUserControl.xaml">
    <SubType>Designer</SubType>
  </Page>
  <Page Update="UserControls\Seances\SeancesUserControl.xaml">
    <SubType>Designer</SubType>
  </Page>
  <Page Update="UserControls\Seances\SingleSeanceUserControl.xaml">
    <SubType>Designer</SubType>
```

```
    </Page>
  </ItemGroup>
</Project>
using CinemaTickets.Authentication;
using CinemaTickets.Models;
using CinemaTickets.Repositories;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.DependencyInjection;
using System.Windows;

namespace CinemaTickets
{
    /// <summary>
    /// Interaction logic for App.xaml
    /// </summary>
    public partial class App : Application
    {
        private ServiceProvider serviceProvider;

        public App()
        {
            ServiceCollection services = new();
            ConfigureServices(services);
            serviceProvider = services.BuildServiceProvider();
        }

        private void ConfigureServices(ServiceCollection services)
        {
            services.AddDbContext<CinematicketsContext>(options => options.UseSqlServer("Data Source=loc
            services.AddSingleton<MainWindow>();

            services.AddScoped<IClientRepository, ClientRepository>();
            services.AddScoped<IEmployeeRepository, EmployeeRepository>();
            services.AddScoped<IHallRepository, HallRepository>();
            services.AddScoped<IMovieRepository, MovieRepository>();
            services.AddScoped<ISeanceRegistrationRepository, SeanceRegistrationRepository>();
            services.AddScoped<ISeanceRepository, SeanceRepository>();

            services.AddScoped<IPasswordCryption, PasswordCryption>();
            services.AddScoped<IAuthStore, AuthStore>();
            services.AddScoped<IAuthService, AuthService>();
        }

        private void OnStartup(object sender, StartupEventArgs e)
        {
            var mainWindow = serviceProvider.GetService<MainWindow>();
            mainWindow?.Show();
        }
    }
}

<Window x:Class="CinemaTickets.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

```
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:CinemaTickets"
        mc:Ignorable="d"
        Title="Cinema Tickets" Height="450" Width="800">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition MinHeight="35" MaxHeight="40"/>
            <RowDefinition Height="9*"/>
        </Grid.RowDefinitions>
        <DockPanel Grid.Row="0" LastChildFill="False">
            <StackPanel x:Name="NavigationStackPanel" Orientation="Horizontal"/>
            <Button Grid.Row="0" x:Name="LogoutButton" Visibility="Hidden" MinWidth="140" DockPanel.Dock
                    Click="OnLogoutClick"/>
        </DockPanel>
        <ContentControl Grid.Row="1" x:Name="MainContentControl"/>
    </Grid>
</Window>

using System.Windows;

[assembly: ThemeInfo(
    ResourceDictionaryLocation.None, //where theme specific resource dictionaries are located
                                     //(used if a resource is not found in the page,
                                     // or application resource dictionaries)
    ResourceDictionaryLocation.SourceAssembly //where the generic resource dictionary is located
                                              //(used if a resource is not found in the page,
                                              // app, or any theme specific resource dictionaries)
)]
```