*Tytuł projektu:*

**System rezerwacji seansów w kinie studyjnym Cinematrix**

*Krótki opis działania projektu:*

**Aplikacja umożliwia użytkownikom kina przeglądanie dostępnych seansów, rezerwowanie miejsc na seanse oraz przeglądanie aktywnych rezerwacji. Administratorzy mogą zarządzać harmonogramem seansów, dodawać i usuwać seanse, a także monitorować obłożenie. System posiada podział na role użytkowników np. obserwator, użytkownik standardowy, administrator.**

*Autorzy projektu:*

**Kamil Markiewicz**
**Kacper Białowąs**

1) Specyfikacja wykorzystanych technologii:
   W projekcie zastosowano technologię C# .NET 8 w połączeniu z bazą danych SQL Server, co zapewnia wydajną i bezpieczną obsługę danych.
   Do komunikacji z bazą danych użyliśmy technologii ORM Entity Framework, co umożliwia mapowanie obiektów na tabele bazy danych i upraszcza pracę z danymi dzięki operacjom CRUD realizowanym bezpośrednio na modelach aplikacji.
   W projekcie zastosowano Entity Framework w celu zwiększenia efektywności pracy z bazą danych i zmniejszenia ilości ręcznie pisanych zapytań SQL.
   W projekcie zastosowano podejście *code-first* Entity Framework, które umożliwia definiowanie struktury bazy danych bezpośrednio na podstawie klas modelowych w kodzie aplikacji.
   Podejście *code-first* pozwala na automatyczne generowanie schematu bazy danych z poziomu kodu dzięki migracjom, które są uruchamiane za pomocą komendy update-database w Menedżerze NuGet lub dotnet ef database update w terminalu.
   Zastosowanie podejścia *code-first* upraszcza proces tworzenia i utrzymywania struktury bazy danych, umożliwiając łatwe aktualizacje poprzez dodawanie zmian w kodzie oraz ich synchronizację z bazą danych.
   Mechanizm migracji Entity Framework pozwala na śledzenie zmian w modelach danych i aktualizowanie struktury bazy danych bez konieczności ręcznego tworzenia skryptów SQL.

2) Instrukcja pierwszego uruchomienia:
    1. Wybrany folder na komputerze należy otworzyć w terminalu.
    2. W terminalu należy wpisać komendę:
       *git clone https://github.com/melonkamil/cinematrix.git*
    3. W powstałym folderze należy uruchomić plik *Cinematrix.sln*.
    4. Po uruchomieniu Visual Studio należy w *Package Manager Console* wykonać komendę *update-database*.
    5. Po wykonaniu powyższych kroków można uruchomić projekt.
    6. W oknie przeglądarki otworzy się aplikacja do przeglądania, zarządzania i rezerwacji seansów w kinie Cinematrix.
    7. Baza danych jest pusta, należy zalogować się kontem administratora
       *admin@cinematrix.com*
       *P@ssw0rd*
       celem dodania nowych filmów i seansów.
    8. Dodając film należy podać tytuł, opis oraz link URL do plakatu.
    9. Dodając seans należy wybrać film, podać datę oraz godzinę zgodnie z podanym schematem w podpowiedzi:
       -data RRRR-MM-DD
       -godzina GG:MM

3) Opis struktury projektu:
    a) Modele w projekcie:
    1. ApplicationUser
       - Pełni rolę rozszerzenia domyślnego modelu użytkownika w systemie uwierzytelniania. Jest częścią mechanizmu ASP.NET Core Identity, który zapewnia gotowe narzędzia do obsługi logowania, rejestracji, zarządzania rolami użytkowników itp.
       - Pola:
       public virtual TKey Id { get; set; } = default!;
       unikalny identyfikator użytkownika

       public virtual string? Email { get; set; }
       adres e-mail

       public virtual string? PasswordHash { get; set; }
       hasz hasła

2. Movie
   - Reprezentuje model danych dla filmów w aplikacji. Jest to klasa, która definiuje strukturę informacji, jakie aplikacja przechowuje na temat filmów, np. w bazie danych. Służy ona do przechowywania i zarządzania danymi o filmach, które mogą być wyświetlane użytkownikom.
   - Pola:
   public string Id { get; set; }
   to unikalny identyfikator filmu w bazie danych

   public string Title { get; set; }
   to unikatowa nazwa filmu w systemie, posiada ograniczenie do 200 znaków

   public string Description { get; set; }
   służy do przechowywania dodatkowych informacji opisowych o danym filmie, posiada ograniczenie do 1000 znaków

   public string ImageUrl { get; set; }
   służy do załączenia obrazu plakatu filmu z sieci, dzięki czemu nie ma potrzeby przechowywania dużego pliku w bazie danych, pole posiada ograniczenie do 500 znaków

   public bool IsDeleted { get; set; }
   znacznik logiczny na potrzeby miękkiego usunięcia filmu z bazy danych, a dokładnie do usuwania z widoku na stronie rezerwacji

   public DateTime CreatedAt { get; set; }
   znacznik czasowy na potrzeby przechowania informacji o dacie i godzinie utworzenia wpisu o filmie

3. Reservation
   - Służy do reprezentowania rezerwacji miejsca na seans filmowy. Jest to model danych opisujący szczegóły rezerwacji dokonanej przez użytkownika, takie jak użytkownik, wybrany seans, koszt rezerwacji oraz czas jej utworzenia.
   - Pola:
   public string Id { get; set; }
   to unikalny identyfikator rezerwacji w bazie danych

   public string UserId { get; set; }
   to unikalny identyfikator użytkownika w bazie danych

   public string ShowtimeId { get; set; }
   to unikalny identyfikator seansu w bazie danych

   public int Cost { get; set; }
   służy do wyliczania sumarycznego kosztu rezerwacji

   public DateTime CreatedAt { get; set; }
   znacznik czasowy na potrzeby przechowania informacji o dacie i godzinie utworzenia rezerwacji

   public Showtime Showtime { get; set; }
   służy do przechowania informacji o seansie, którego dotyczy rezerwacja

4. ReservedSeat
   - Służy do reprezentowania informacji o konkretnym zarezerwowanym miejscu na dany seans filmowy. W aplikacji jest ona wykorzystywana do przechowywania danych dotyczących tego, które miejsca na danym seansie zostały zarezerwowane przez użytkowników.
   - Pola:
   public string Id { get; set; }
   to unikalny identyfikator miejsca w bazie danych

   public string ShowtimeId { get; set; }
   to unikalny identyfikator seansu w bazie danych

   public string ReservationId { get; set; }
   to unikalny identyfikator rezerwacji w bazie danych

   public string SeatNumber { get; set; }
   to numer miejsca wskazywanego przez użytkownika

   public DateTime CreatedAt { get; set; }
   znacznik czasowy na potrzeby przechowania informacji o dacie i godzinie

5. Showtime
   - Służy do reprezentowania konkretnego seansu filmowego w systemie rezerwacji biletów. W aplikacji jest używana do przechowywania informacji o dacie, godzinie oraz filmie wyświetlanym na danym seansie. Dzięki tej klasie można zarządzać harmonogramem wyświetlania filmów.
   - Pola:
   public string Id { get; set; }
   to unikalny identyfikator seansu w bazie danych

   public string MovieId { get; set; }
   to unikalny identyfikator filmu w bazie danych

   public string Date { get; set; }
   służy do przechowania informacji o dacie seansu

   public string Time { get; set; }
   służy do przechowania informacji o godzinie rozpoczęcia seansu

   public bool IsDeleted { get; set; }
   znacznik logiczny na potrzeby miękkiego usunięcia seansu z bazy danych, a dokładnie do usuwania z widoku na stronie rezerwacji

   public DateTime CreatedAt { get; set; }
   znacznik czasowy na potrzeby przechowania informacji o dacie i godzinie utworzenia seansu

   public Movie Movie { get; set; }
   służy do przechowania informacji o filmie jakiego dotyczy seans

6. ReservationView
   - Pełni rolę modelu widoku (ViewModel) w aplikacji. Służy do przekazywania danych dotyczących rezerwacji z warstwy serwisowej lub kontrolera do widoku (interfejsu użytkownika). Umożliwia prezentację informacji o rezerwacji w formacie przyjaznym dla użytkownika, łącząc dane z różnych tabel/modeli.
   - Pola:
   public string ReservationId { get; set; }
   służy do wyświetlenia identyfikatora rezerwacji

   public string MovieTitle { get; set; }
   służy do wyświetlenia tytułu filmu rezerwacji

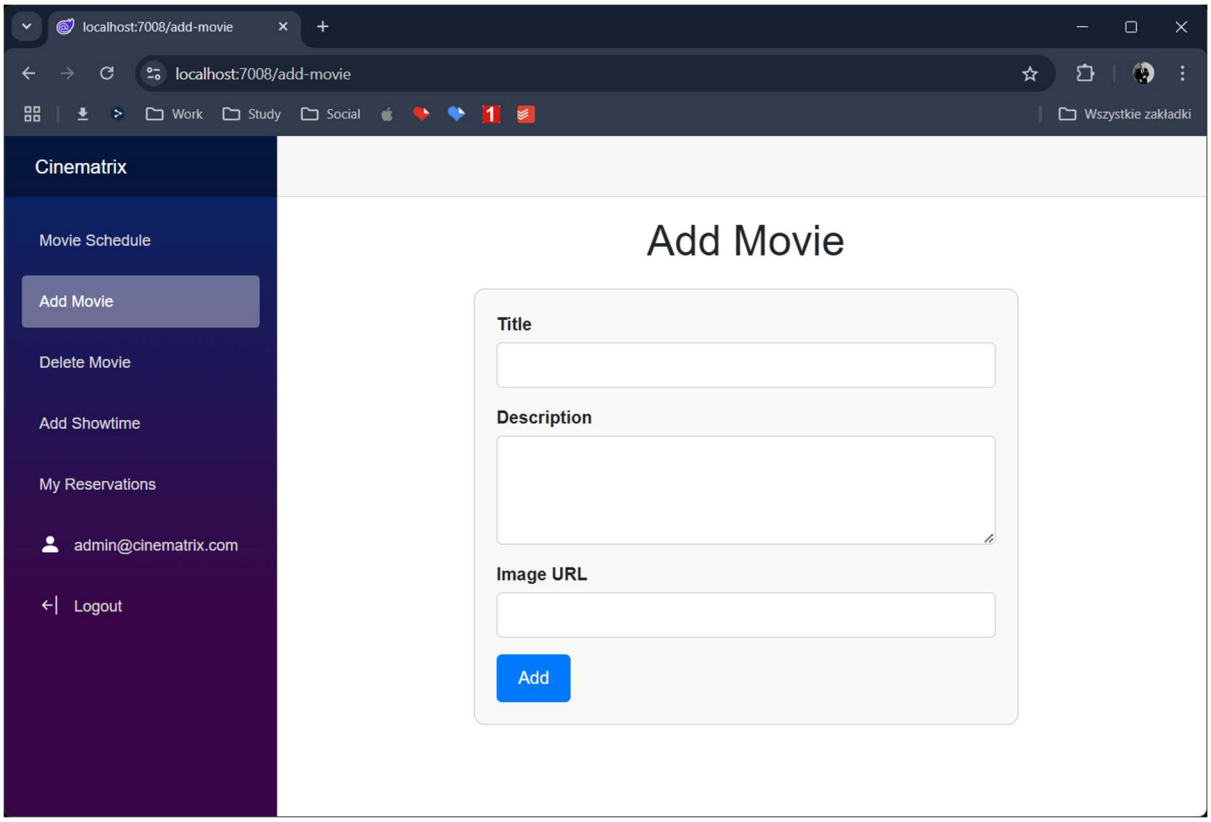
   public string Date { get; set; }
   służy do wyświetlenia daty rezerwacji

   public string Time { get; set; }
   służy do wyświetlenia godziny rezerwacji

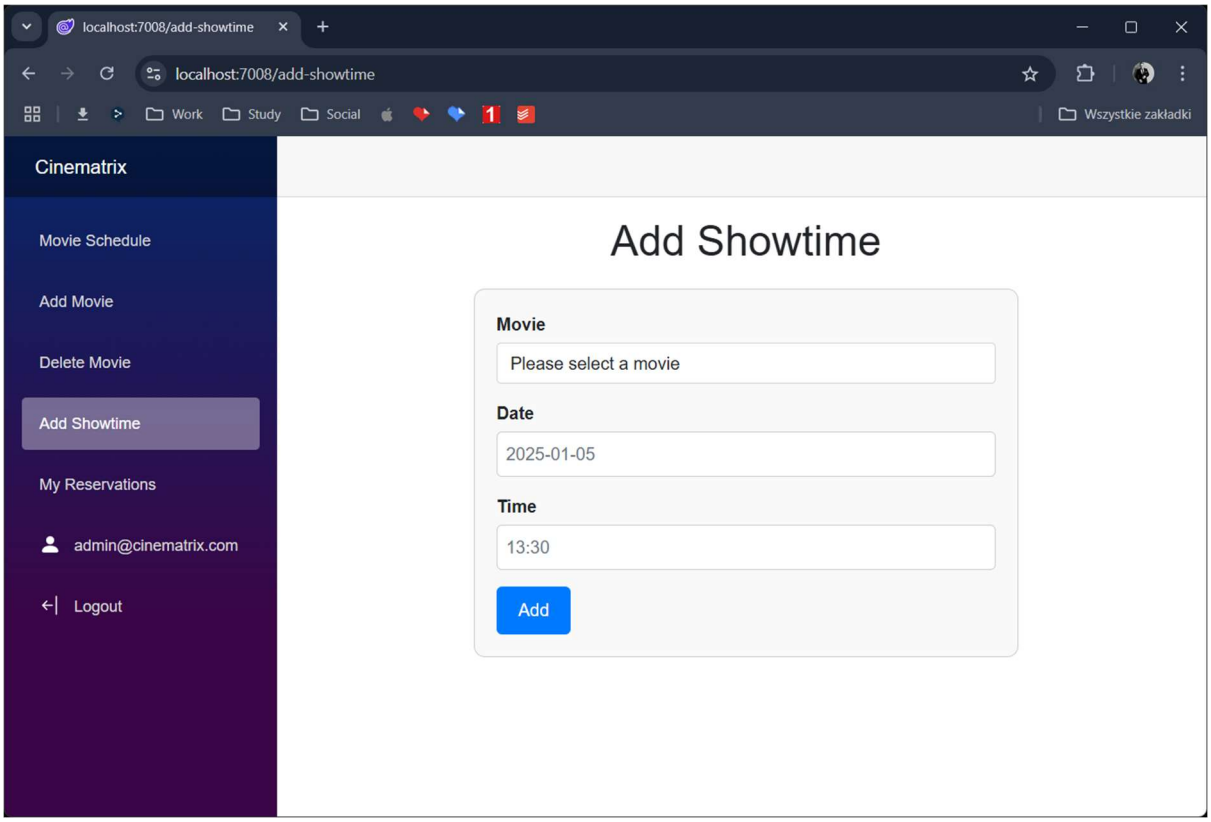
   public int TotalCost { get; set; }
   służy do wyświetlenia sumarycznego kosztu rezerwacji
   public string[] Seats { get; set; }
   służy do wyświetlenia zarezerwowanych miejsc

b) Strony: (URL)

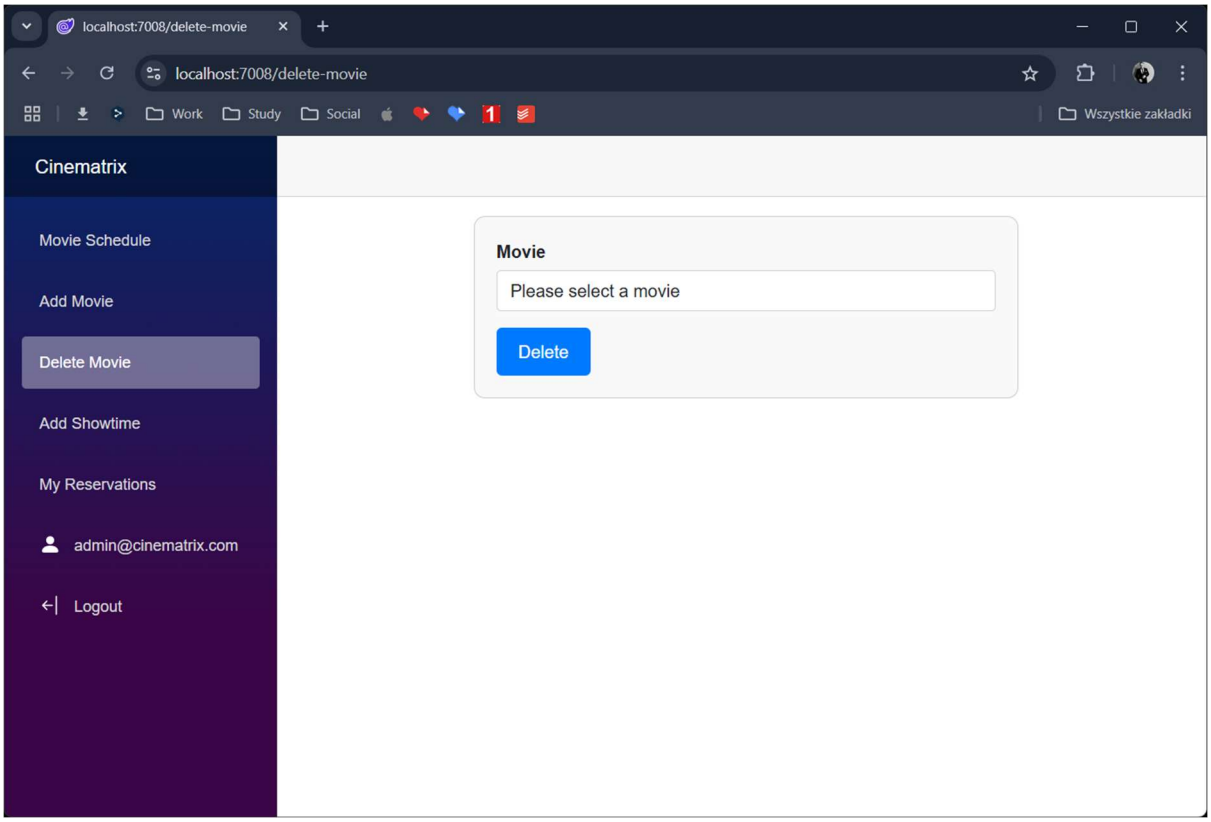
1. AddMovie ("/add-movie")
   Strona dostępna tylko dla administratora dzięki *@attribute [Authorize(Policy = "ManageMoviesPolicy")]*
   Służy do dodawania nowego filmu w aplikacji. Komponent obsługuje formularz zawierający pola: tytuł filmu, opis oraz adres URL obrazu, a także zapewnia walidację danych i wyświetlanie komunikatów o błędach. Dodatkowo, po poprawnym wypełnieniu formularza, użytkownik musi potwierdzić dodanie filmu w oknie pop-up.



2. AddShowtime ("/add-showtime")
   Strona dostępna tylko dla administratora dzięki *@attribute [Authorize(Policy = "ManageMoviesPolicy")]*
   Służy do dodawania nowego seansu filmowego. Pozwala na wybranie filmu z listy dostępnych tytułów, a następnie podanie daty i godziny seansu. Komponent obsługuje walidację danych, wyświetlanie komunikatów o błędach oraz potwierdzenie operacji przed dodaniem seansu do systemu.

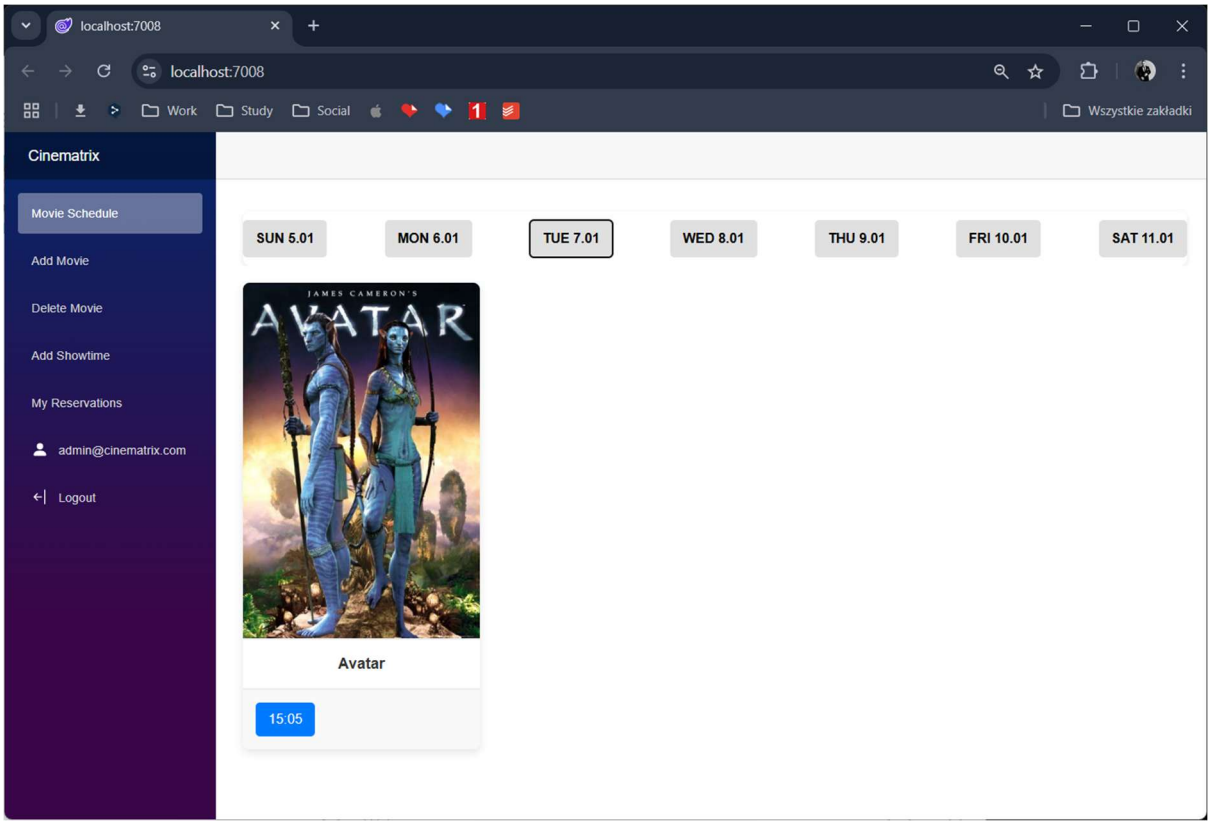


3. DeleteMovie ("/delete-movie")
   Strona dostępna tylko dla administratora dzięki
   *@attribute [Authorize(Policy = "ManageMoviesPolicy")]*
   Umożliwia usuwanie filmu wraz z jego seansami z bazy danych. Użytkownik może wybrać film z listy, a następnie po zatwierdzeniu usunąć go z systemu. Komponent zapewnia walidację danych, wyświetlanie komunikatów o błędach oraz okno potwierdzenia przed usunięciem filmu.



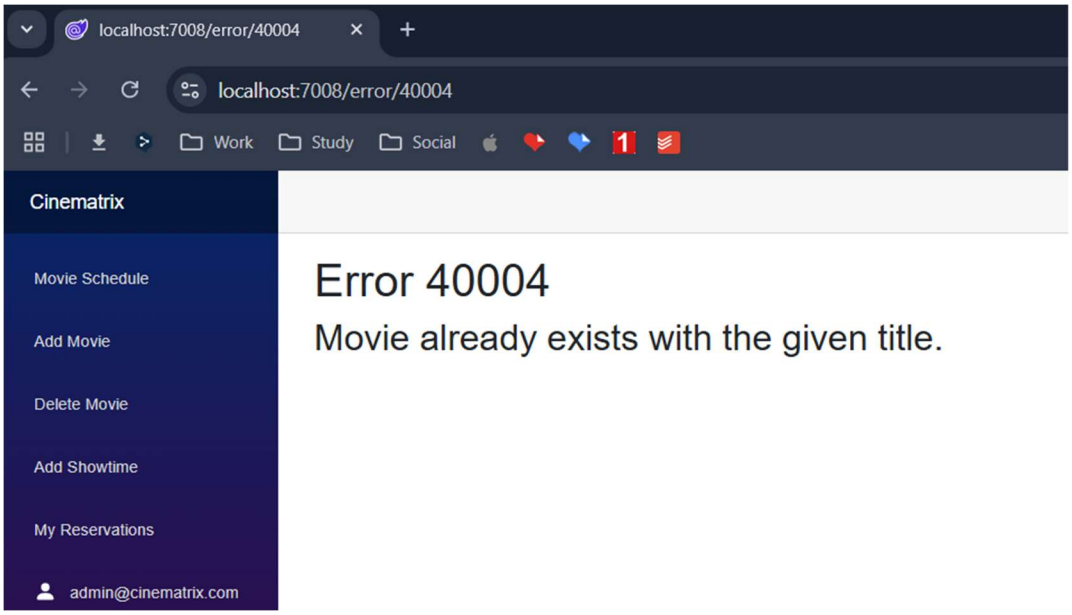

4. Error ("/Error")
   Strona dostępna dla każdego, również niezalogowanego.
   Domyślnie utworzona w projekcie strona, która pełni rolę strony błędu, jest wyświetlana, gdy w aplikacji wystąpi nieoczekiwany błąd. Jest to strona obsługi błędów, która przekazuje użytkownikowi podstawowe informacje o błędzie w bezpieczny i czytelny sposób, unikając wyświetlania szczegółów technicznych, które mogłyby ujawniać wrażliwe informacje.

5. Index ("/")
   Strona dostępna dla każdego, również niezalogowanego.
   Pełni rolę strony głównej aplikacji wyświetlającej listę dostępnych filmów oraz godzin seansów dla wybranego dnia. Użytkownik może przełączać dni tygodnia, aby zobaczyć repertuar na wybrany dzień, kliknąć na film, aby zobaczyć szczegóły, lub wybrać godzinę seansu, aby przejść do rezerwacji biletu.

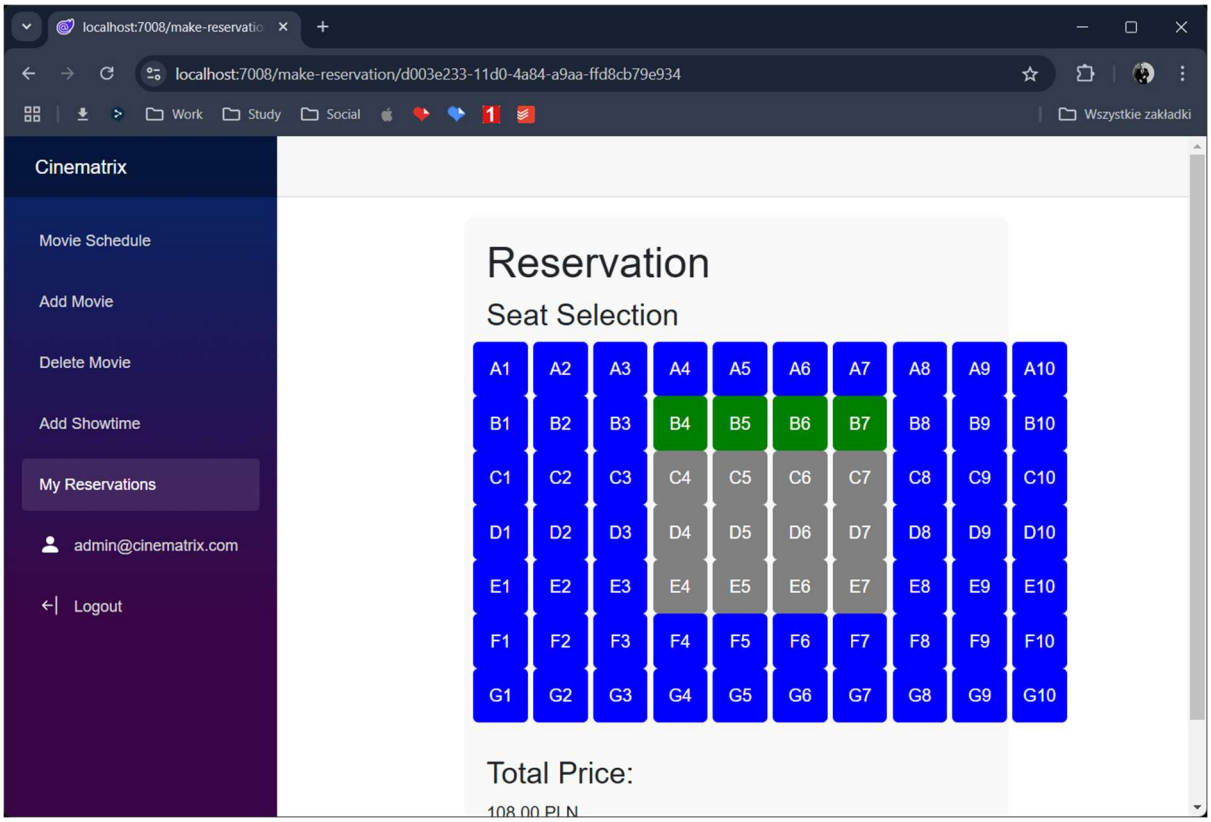


6. Issue ("/error/{ErrorCode}")
   Strona dostępna dla każdego, również niezalogowanego.
   Utworzona na rzecz projektu strona, podobna do „Error”, która pełni rolę wyświetlania znanych, spodziewanych błędów z logiki biznesowej.
   Na tę stronę zostaje przeniesiony użytkownik w sytuacji wystąpienia błędu poziomu biznesowego, np.: chcąc dodać film, który już istnieje lub w momencie wystąpienia błędów logiki biznesowej, np.: kiedy dodając seans zostanie użyty nieprawidłowy pattern daty lub godziny.
   Powiązane kody błędów:

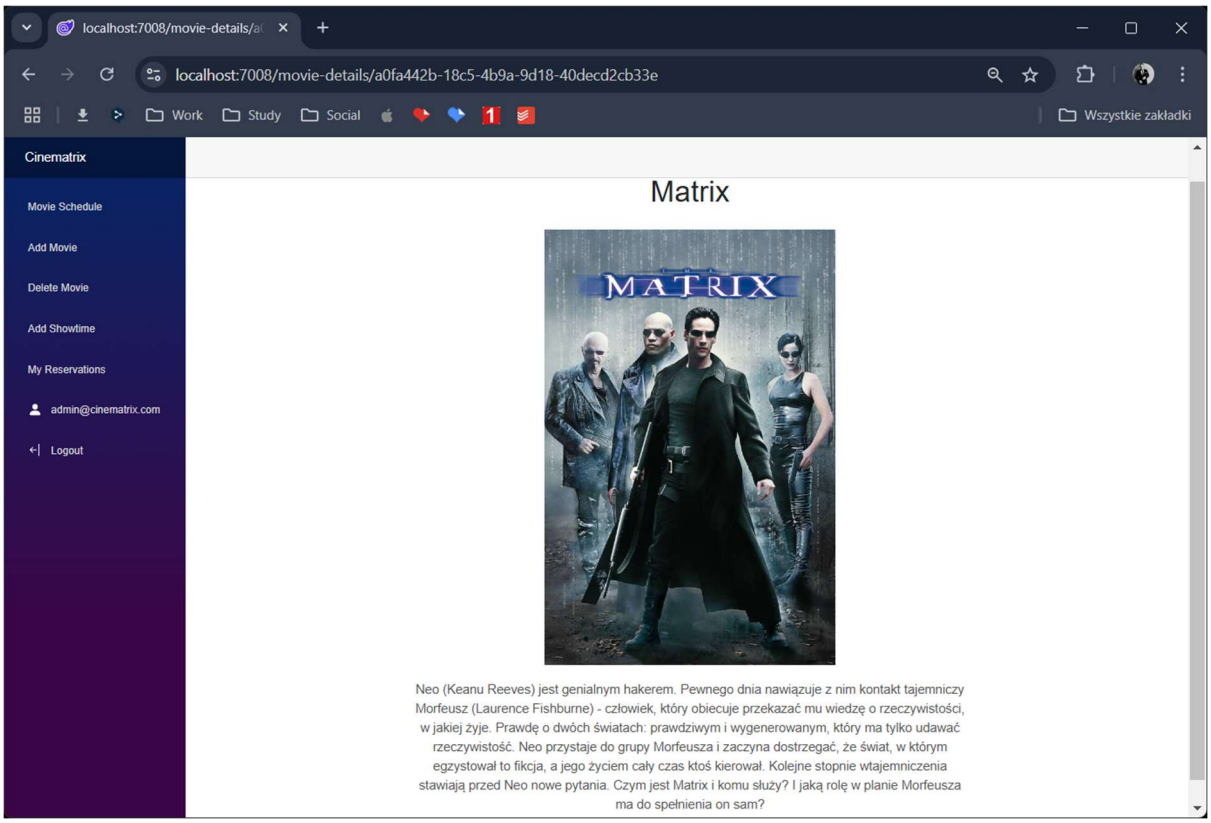
```
ErrorCodeRegistry.cs
Cinematrix | Cinematrix.Services.ErrorCodeRegistry | ErrorMessageByCode

using System.Collections.Immutable;

namespace Cinematrix.Services;

1 reference
public static class ErrorCodeRegistry
{
    public static readonly ImmutableDictionary<int, string> ErrorMessageByCode =
        new Dictionary<int, string>
        {
            [40001] = "Movie title is invalid",
            [40002] = "Movie description is invalid",
            [40003] = "Movie image url is invalid",
            [40004] = "Movie already exists with the given title",
            [40005] = "Movie not found",
            [40006] = "At least one showtime reservation is purchased",
            [40007] = "Showtime date is invalid",
            [40008] = "Showtime time is invalid",
            [40009] = "Showtime already exists",
            [40010] = "Showtime not found",
            [40011] = "Seats are not valid",
            [40012] = "Seats already reserved",
            [40401] = "Movie not found",
            [40402] = "Showtime not found",
        }.ToImmutableDictionary();
}
```



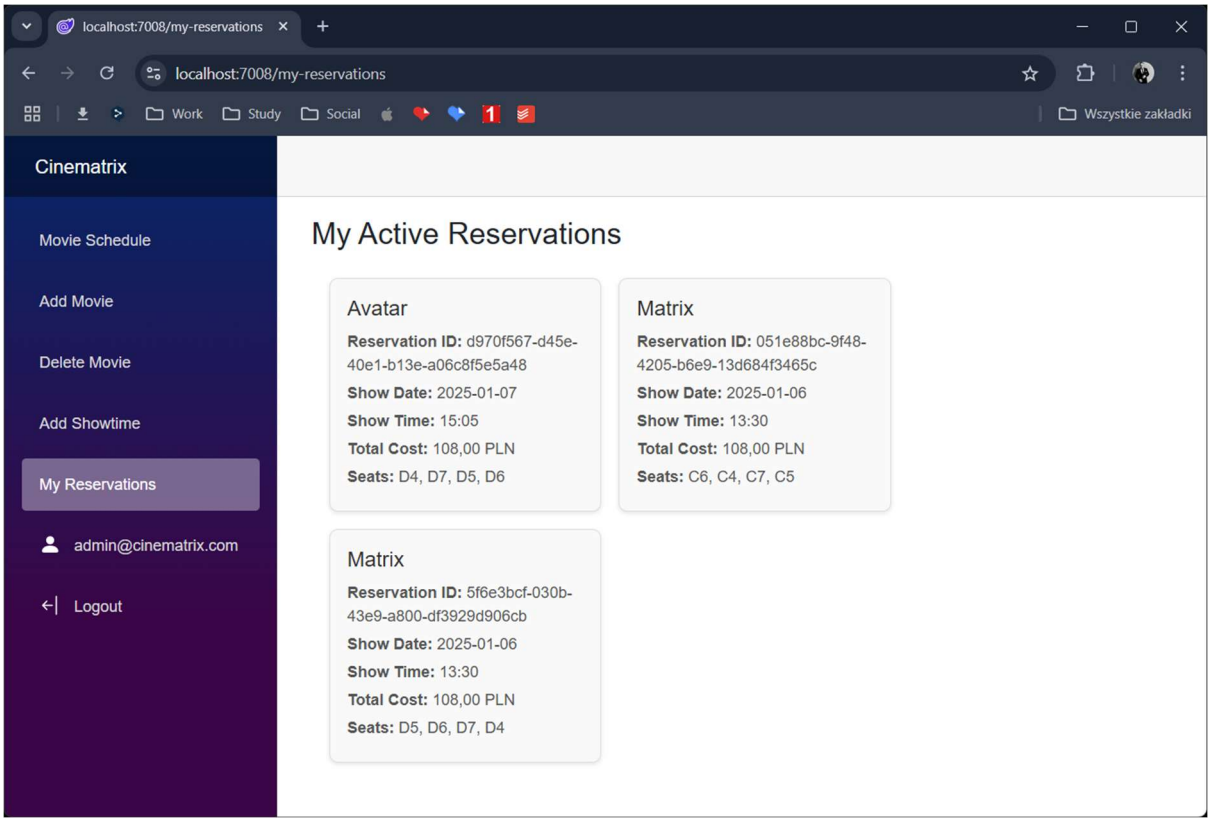
7. MakeReservation ("/make-reservation/{ShowtimeId}")
   Strona dostępna tylko dla zalogowanego użytkownika dzięki *@attribute [Authorize]*
   Umożliwia użytkownikowi rezerwację miejsc na wybrany seans filmowy. Po wejściu na stronę /make-reservation/{ShowtimeId} użytkownik widzi układ sali kinowej, wybiera miejsca, a następnie potwierdza rezerwację. Komponent obsługuje walidację wybranych miejsc, wyświetla łączny koszt biletów i umożliwia potwierdzenie lub anulowanie zakupu.



8. MovieDetails ("/movie-details/{MovieId}")
   Strona dostępna dla każdego, również niezalogowanego.
   Służy do wyświetlania szczegółów wybranego filmu. Po wejściu na stronę /movie-details/{MovieId}, gdzie MovieId to identyfikator filmu, użytkownik widzi tytuł, plakat oraz opis filmu.



9. MyReservations ("/my-reservations")
   Strona dostępna tylko dla zalogowanego użytkownika dzięki *@attribute [Authorize]*
   Wyświetla listę aktywnych rezerwacji użytkownika. Po zalogowaniu użytkownik może przejść na stronę /my-reservations, aby zobaczyć szczegóły swoich aktualnych rezerwacji, takich jak tytuł filmu, data i godzina seansu, cena oraz wybrane miejsca.



4) Metody

a) FindMovie

Opis: Wyszukuje film na podstawie identyfikatora `movieId`.
Parametry:
- `movieId` (string) – identyfikator filmu.
- `ct` (CancellationToken) – opcjonalny token anulowania.
Walidacje: Sprawdza, czy film o podanym identyfikatorze istnieje.
Błąd: `NotFoundException (40401)` – jeśli film nie został znaleziony.
Zwracane dane:** Obiekt `Movie`.

b) GetUserActiveReservations

Opis: Pobiera aktywne rezerwacje użytkownika
(seanse, które jeszcze się nie odbyły).
Parametry:
- `userId` (string) – identyfikator użytkownika.
- `ct` (CancellationToken) – opcjonalny token anulowania.
Walidacje: Brak.
Błąd: Brak.
Zwracane dane: Lista obiektów `ReservationView`, zawierająca szczegóły rezerwacji (tytuł filmu, data, godzina, cena i miejsca).

c) GetTicketPricePLN`**

Opis: Zwraca cenę biletu w groszach.
Parametry: Brak.
Walidacje: Brak.
Błąd: Brak.
Zwracane dane: `int` – cena biletu (domyślnie `2700` groszy, czyli `27 PLN`).

d) GetReservedSeats

Opis: Pobiera listę zarezerwowanych miejsc dla danego seansu.
Parametry:
- `showtimeId` (string) – identyfikator seansu.
- `ct` (CancellationToken) – opcjonalny token anulowania.
Walidacje: Sprawdza, czy seans istnieje i nie jest oznaczony jako usunięty.
Błąd: Brak – jeśli seans nie istnieje, zwracana jest pusta lista.
Zwracane dane: Lista numerów miejsc (`List<string>`).

e) GetMovie

Opis: Pobiera listę aktywnych (nieusuniętych) filmów.
Parametry:
- `ct` (CancellationToken) – opcjonalny token anulowania.
Walidacje: Brak.
Błąd: Brak.
Zwracane dane: Lista obiektów `Movie`.

f) GetShowtimesByDate

Opis: Pobiera listę seansów dla danego dnia.
Parametry:
- `date` (string) – data w formacie `yyyy-MM-dd`.
- `ct` (CancellationToken) – opcjonalny token anulowania.
Walidacje: Brak.
Błąd: Brak.
Zwracane dane: Lista obiektów `Showtime`.

g) AddMovie

Opis: Dodaje nowy film do bazy danych.
Parametry:
- `title` (string) – tytuł filmu.
- `description` (string) – opis filmu.
- `imageUrl` (string) – URL plakatu filmu.
- `ct` (CancellationToken) – opcjonalny token anulowania.
-Walidacje:
- `title`, `description`, `imageUrl` – nie mogą być puste.
- Sprawdza, czy film o tym samym tytule już istnieje.
Błąd:
- `BadRequestException (40001–40004)` – dla pustych wartości lub duplikatów.
Zwracane dane: `string` – identyfikator nowo dodanego filmu.

h) DeleteMovieAndShowtimes

Opis: Oznacza film i jego seanse jako usunięte, jeśli nie ma dla nich rezerwacji.
Parametry:
- `movieId` (string) – identyfikator filmu.
- `ct` (CancellationToken) – opcjonalny token anulowania.
Walidacje:
- Sprawdza, czy film istnieje i nie jest już oznaczony jako usunięty.
- Sprawdza, czy dla danego filmu istnieją aktywne rezerwacje.
Błąd:
- `BadRequestException (40005)` – jeśli film nie istnieje.
- `BadRequestException (40006)` – jeśli są aktywne rezerwacje.
Zwracane dane: Brak.

i) AddShowtime

Opis: Dodaje nowy seans dla filmu.
Parametry:
- `movieId` (string) – identyfikator filmu.
- `date` (string) – data w formacie `yyyy-MM-dd`.
- `time` (string) – godzina w formacie `HH:mm`.
- `ct` (CancellationToken) – opcjonalny token anulowania.
Walidacje:
- Sprawdza, czy film istnieje.
- Waliduje format daty i godziny za pomocą wyrażeń regularnych:
SHOWTIME_DATE_PATTERN = @"^\d{4}-\d{2}-\d{2}$";
SHOWTIME_TIME_PATTERN = "^(0[0-9]|1[0-9]|2[0-3]):([0-5][0-9])$";
- Sprawdza, czy seans o podanej dacie i godzinie już istnieje.
Błąd:
- `BadRequestException (40005, 40007, 40008, 40009)` – błędy związane z brakiem filmu lub niepoprawnym formatem danych.
Zwracane dane: `string` – identyfikator nowo dodanego seansu.

j) FindShowtime

Opis: Wyszukuje seans na podstawie identyfikatora filmu i daty oraz godziny.
Parametry:
- `movieId` (string) – identyfikator filmu.
- `dateTime` (DateTime) – data i godzina seansu.
- `ct` (CancellationToken) – opcjonalny token anulowania.
Walidacje: Sprawdza, czy seans istnieje i nie jest oznaczony jako usunięty.
Błąd: `NotFoundException (40402)` – jeśli seans nie został znaleziony.
Zwracane dane: Obiekt `Showtime`.

k) MakeReservation

Opis: Tworzy rezerwację na wybrane miejsca dla seansu.

Parametry:

- `showtimeId` (string) – identyfikator seansu.
- `seats` (string[]) – lista numerów miejsc.
- `userId` (string) – identyfikator użytkownika.
- `ct` (CancellationToken) – opcjonalny token anulowania.

Walidacje:

- Sprawdza, czy seans istnieje i nie jest oznaczony jako usunięty.
- Sprawdza, czy miejsca mają poprawny format (np. `A5`).

SEAT_NUMBER_PATTERN = @"^[A-Z]\d{1,2}$";

- Sprawdza, czy podano co najmniej jedno miejsce.
- Sprawdza, czy wybrane miejsca nie są już zarezerwowane.

Błąd:

- `BadRequestException (40010–40012)` – błędy związane z nieistniejącym seansem, pustą listą miejsc lub podwójną rezerwacją.

Zwracane dane: `string` – identyfikator nowej rezerwacji.

Dane zapisywane do bazy:

- Rezerwacja (`Reservation`).
- Lista miejsc (`ReservedSeat`).

5) Opis systemu użytkowników
W naszym projekcie wykorzystaliśmy wbudowany mechanizm uwierzytelniania i autoryzacji oferowany przez platformę Microsoft.
Dzięki użyciu konfiguracji
builder.Services.AddIdentityCore<ApplicationUser>()

oraz
AddRoles<IdentityRole>()

zarejestrowaliśmy usługę Identity, która obsługuje rejestrację, logowanie i zarządzanie użytkownikami oraz rolami.

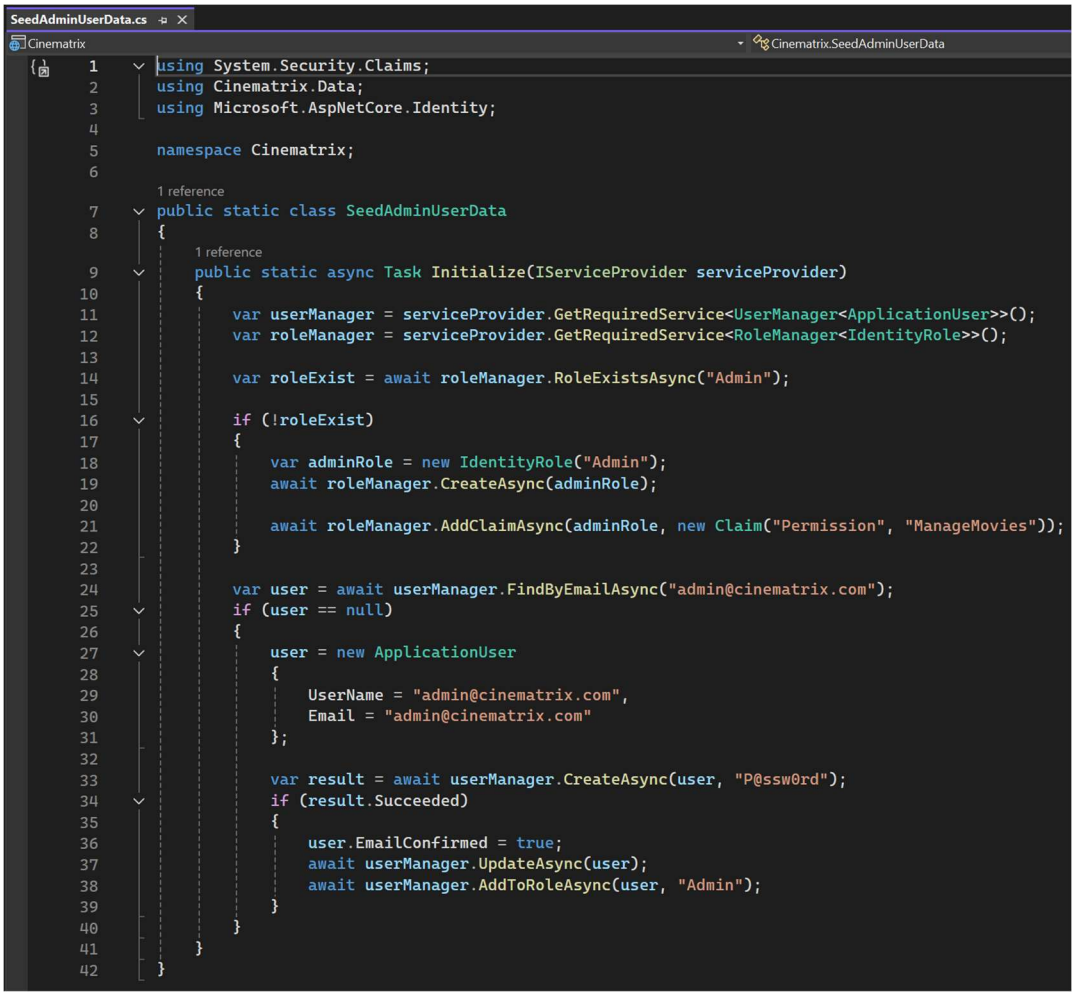
Ustawienie
options.SignIn.RequireConfirmedAccount = true

dodatkowo wymusza, aby użytkownik przed zalogowaniem potwierdził swoje konto, co zwiększa bezpieczeństwo procesu logowania. W efekcie mamy kompleksowe wsparcie dla operacji związanych z tożsamością użytkowników, bez potrzeby implementowania tych mechanizmów od podstaw.

Stworzyliśmy politykę autoryzacji opartą na claimach, co pozwala na precyzyjne kontrolowanie dostępu do stron i operacji w aplikacji. Dzięki temu tylko użytkownicy posiadający odpowiednie uprawnienia mogą np. zarządzać stroną czy wykonywać działania administracyjne. Mechanizm ten zwiększa bezpieczeństwo i umożliwia przypisywanie ról oraz praw dostępu zgodnie z indywidualnymi uprawnieniami użytkowników.

```
builder.Services.AddAuthorization(options =>
{
options.AddPolicy("ManageMoviesPolicy", policy =>
policy.RequireClaim("Permission", "ManageMovies"));
});
```

Podczas uruchamiania systemu automatycznie tworzy się rola Admin, do której przypisywane jest uprawnienie "manage movies". Następnie to uprawnienie jest powiązane z rolą Admin, a konto użytkownika admin@cinematrix.com jest tworzone i przypisywane do tej roli. Cała ta logika jest zaimplementowana w pliku SeedAdminUserData.cs. Jeśli konto admin@cinematrix.com zostanie usunięte, system przy kolejnym uruchomieniu odtworzy je ponownie.




Każdy nowo zarejestrowany użytkownik otrzymuje domyślnie status zwykłego użytkownika bez dodatkowych uprawnień. Uprawnienia mogą być nadane takim użytkownikom wyłącznie ręcznie, poprzez edycję danych w bazie.

Gość (niezalogowany użytkownik) ma jedynie możliwość przeglądania seansów. W momencie wyboru godziny seansu, aplikacja automatycznie przekierowuje go do strony logowania lub rejestracji. Bez zalogowania się lub założenia konta nie można przejść do rezerwacji ani obejść procesu w celu dokonania rezerwacji anonimowo. Taki mechanizm zapobiega nieautoryzowanym operacjom i zabezpiecza system przed próbami obejścia ścieżki autoryzacji.

6) Krótka charakterystyka najciekawszych funkcjonalności
    a) Make reservation - posiada siatkę przedstawiającą wizualny rozkład siedzeń w kinie, które informują użytkownika o dostępnych miejscach. Wolne miejsca są widoczne w kolorze niebieskim, wybrane przez użytkownika są podświetlone na zielono, a zajęte przez innych widnieją na szaro oraz nie są dostępne do zaznaczenia.
    b) Delete movie and showtimes – funkcja która przy usunięciu wybranego filmu w repertuarze usuwa również wszystkie seanse tego filmu co pozwala na szybkie zarządzanie bez wymogu usuwania każdego seansu z osobna.
    c) Get user active reservation – funkcja która pobiera dane o rezerwacjach zalogowanego użytkownika i wyświetla tylko je w podglądzie rezerwacji przez co każdy użytkownik widzi tylko rezerwacje wykonane przez jego konto i nie może ingerować w inne.