Dokumentacja Projektu Rezerwowania miejsc w kinie:

Posiadane przez nas klasy:

- Kino – Przechowuje informacje o kliencie.

a) string imie,

b) string nazwisko,

c) string telefon,

d) string mail.

Do każdego pola we właściwościach mamy dodanie sprawdzania czy imię i nazwisko składa się z samych liter, czy telefon składa się z 9 cyfr, oraz czy mail ma format tekst@tekst.tekst.

- MovieInfo – Przechowuje informacje o filmie.

a) string title,

b) stringe imagePath,

c) string description,

d) string director,

e) string duration,

f) string genre

- Sala – Przechowuje listę przycisków z miejscami.

a) List<string> listaPrzyciskow

- RezerwacjaNowa – Przechowuje informacje o rezerwacji.

a) static int index,

b) string numerRezerwacji,

c) Sala sala,

d) string zarezerwowaneMiejsca,

e) decimal cena,

f) Klient klient,

g) bool oplacona

Interfejsy:

- ISerialize1 – Służy do serializacji do XML.

void SaveToXml(string file) – zapis do XML

GUI:

- MainWindow – Okno główne, na którym znajduje się nazwa kina oraz przycisk do sprawdzenia repertuaru.

Funkcje:

- Przechodzenie do kolejnej strony.

```
private void Repertuar_Click(object sender, RoutedEventArgs e)
    {
      RepertuarKina repertuar = new RepertuarKina();
// Używamy NavigationService.Navigate aby przejść do kolejnej strony.
      MainFrame.NavigationService.Navigate(repertuar);
    }
```

- RepertuarKina – Strona na której znajdują się dostępne filmy. Umożliwia wybranie filmu, którego specyfikację chcemy zobaczyć.

Funkcje:

- Przechodzenie do specyfikacji filmu i dodanie informacji o filmach.

```
    private void Obrazek_Click(object sender, MouseButtonEventArgs e)
        {
    // Sender to obiekt, który wywołał zdarzenie, czyli kliknięty obrazek.
          Image clickedImage = (Image)sender;
    // Lista z informacjami o filmach.
          List<MovieInfo> movies = new List<MovieInfo>
          {
            new MovieInfo
            {
              Title = "Zootopia",
              ImagePath = "/Images/s-l1200.jpg",
```

```
            Description = "Zwierzogród to jedyne miasto zamieszkiwane wyłącznie przez zwierzęta. Tu możesz zostać kimkolwiek chcesz.\n" +
            "Jednak ambitna Judy Hops, szybko przekonuje się, że jako pierwszy królik w policji, nie będzie miała łatwego życia.\n" +
            "Aby udowodnić swoją wartość musi rozwiązać pewną kryminalną zagadkę. Jej partnerem w śledztwie zostaje gadatliwy i szczwany lis Nick Bajer.",
            Director = " Rich Moore, Byron Howard",
            Duration = "108 minut",
            Genre = "komedia kryminalna, przygodowy"
        },
// Informacje o pozostałych filmach.
    };

// Utworzenie nowego obiektu klasy SpecyfikacjaFilmu, przekazanie informacji o filmie i nawigacja do strony specyfikacji filmu
    SpecyfikacjaFilmu movieDetailsPage = new SpecyfikacjaFilmu(movies[MainStackPanel.Children.IndexOf(clickedImage)]);
    NavigationService.Navigate(movieDetailsPage);
}
```

- Zamknięcie programu.

```
private void Zamknij_Click(object sender, RoutedEventArgs e)
{
    // Zamknięcie aplikacji
    Application.Current.Shutdown();
}
```

- SpecyfikacjaFilmu – Strona zawierająca informacje o wybranym filmie oraz przycisk umożlwiający przejście do zakupu biletów.

Funkcje:

- Konstruktor klasy SpecyfikacjaFilmu, który jest wywoływany podczas tworzenia instancji tej klasy. Przyjmuje on obiekt MovieInfo jako argument, inicjalizuje pole movieInfo i ustawia go jako kontekst danych dla tej strony. Kontekst danych jest używany do przekazywania informacji do elementów interfejsu użytkownika.

```
public SpecyfikacjaFilmu (MovieInfo movieInfo)
    {
      InitializeComponent();
      this.movieInfo = movieInfo;
      DataContext = this.movieInfo;
    }
```

- Przejście do strony wyboru seansu.

```
private void KupBilet_Click(object sender, RoutedEventArgs e)
    {
      Wybor wybor = new Wybor();
      this.NavigationService.Navigate(wybor);
    }
```

- Powrót do poprzedniej strony.

```
private void Powrot_Click(object sender, RoutedEventArgs e)
    {
      this.NavigationService.GoBack();
    }
```

- Wybor – Strona umożliwiająca wybór filmu, dnia tygodnia, godziny.

Funkcje:

- Konstruktor klasy Wybor

```
public Wybor()
  {
     InitializeComponent();

     // Ukrycie przycisku "Dalej".
     BtnDalej.Visibility = Visibility.Collapsed;

     // Ustawienie trybu wyboru jednego elementu w liście filmów.
     FilmListBox.SelectionMode = SelectionMode.Single;

     // Dodanie obsługi zdarzeń dla listy wyboru filmów, listy wyboru dni tygodnia oraz listy wyboru godzin.
     FilmListBox.SelectionChanged += FilmListBox_SelectionChanged;
     DzienTygodniaComboBox.SelectionChanged += DzienTygodniaComboBox_SelectionChanged;

     // Dodanie godzin dla poszczególnych dni tygodnia.
     DodajGodziny("Poniedziałek", "10:00", "15:00", "20:00");
     DodajGodziny("Wtorek", "09:00", "16:00", "21:00");
     DodajGodziny("Środa", "15:00", "18:00", "22:00");
     DodajGodziny("Czwartek", "13:00", "17:00", "21:00");
     DodajGodziny("Piątek", "10:00", "14:00", "18:00");

     // Dodanie obsługi zdarzenia zmiany wyboru godziny.
     GodzinaComboBox.SelectionChanged += GodzinaComboBox_SelectionChanged;
  }
```

- Obsługa zdarzenia zmiany wyboru filmu.

```
private void FilmListBox_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    // Wyświetlenie elementów interfejsu użytkownika odpowiedzialnych za wybór dnia tygodnia.
    DzienTygodniaLabel.Visibility = Visibility.Visible;
    DzienTygodniaComboBox.Visibility = Visibility.Visible;

    // Zresetowanie wyboru dnia tygodnia.
    DzienTygodniaComboBox.SelectedIndex = -1;

    // Ukrycie elementów interfejsu użytkownika odpowiedzialnych za wybór godziny i przycisku "Dalej".
    GodzinaLabel.Visibility = Visibility.Collapsed;
    GodzinaComboBox.Visibility = Visibility.Collapsed;
    BtnDalej.Visibility = Visibility.Collapsed;
}
```

- Obsługa zdarzenia zmiany wyboru dnia tygodnia.

```
private void DzienTygodniaComboBox_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    // Wyświetlenie elementów interfejsu użytkownika odpowiedzialnych za wybór godziny.
    GodzinaLabel.Visibility = Visibility.Visible;
    GodzinaComboBox.Visibility = Visibility.Visible;

    // Pobranie nazwy wybranego dnia tygodnia.
    string wybranyDzien = ((ComboBoxItem)DzienTygodniaComboBox.SelectedItem).Content.ToString();

    // Ustawienie godzin dla wybranego dnia tygodnia.
    UstawGodzinyDlaDniaTygodnia(wybranyDzien); }
```

- Obsługa zdarzenia zmiany wyboru godziny.

```
private void GodzinaComboBox_SelectionChanged(object sender,
SelectionChangedEventArgs e)
{
    // Sprawdzenie, czy wybrano godzinę i ustawienie widoczności przycisku "Dalej".
    if (GodzinaComboBox.SelectedIndex != -1)
    {
        BtnDalej.Visibility = Visibility.Visible;
    }
    else
    {
        BtnDalej.Visibility = Visibility.Collapsed;
    }
}
```

- Ustawienie godzin w liście wyboru godzin dla danego dnia tygodnia.

```
private void UstawGodzinyDlaDniaTygodnia(string dzienTygodnia)
{
    // Ustawienie godzin w liście wyboru godzin dla danego dnia tygodnia.
    switch (dzienTygodnia)
    {
        case "Poniedziałek":
            UstawGodzinyComboBox("10:00", "15:00", "20:00");
            break;

        case "Wtorek":
            UstawGodzinyComboBox("09:00", "16:00", "21:00");
            break;

        case "Środa":
            UstawGodzinyComboBox("15:00", "18:00", "22:00");
```

```
            break;

        case "Czwartek":
            UstawGodzinyComboBox("13:00", "17:00", "21:00");
            break;

        case "Piątek":
            UstawGodzinyComboBox("10:00", "14:00", "18:00");
            break;

        default:
            break;
    }

    // Ukrycie przycisku "Dalej".
    BtnDalej.Visibility = Visibility.Collapsed;
}
```

- Dodanie godzin do listy wyboru godzin.

```
private void DodajGodziny(params string[] godziny)
{
    foreach (var godzina in godziny)
    {
        GodzinaComboBox.Items.Add(godzina);
    }
}
```

- Ustawienie godzin w liście wyboru godzin.

```
private void UstawGodzinyComboBox(params string[] godziny)
{
   GodzinaComboBox.Items.Clear();
   DodajGgodziny(godziny);
}
```

- Obsługa przycisku Dalej.

```
private void Dalej_Click(object sender, RoutedEventArgs e)
{
   // Utworzenie obiektów reprezentujących sale i wybór miejsc.
   Sala sala = new Sala();
   WyborMiejsca wybor = new WyborMiejsca(sala);

   // Nawigacja do kolejnej strony.
   this.NavigationService.Navigate(wybor);
}
```

- Metoda inicjująca obsługę przycisków w sali kinowej

```
   private void PrzygotujPrzyciski()
   {
      for (int i = 1; i <= 50; i++)
      {
         Button przycisk = FindName($"Przycisk{i}") as Button;

         if (przycisk != null)
         {
            przycisk.Click += WyborMiejsca_Przyciski;
         }
      }
   }
```

- Obsługa kliknięcia przycisku w sali kinowej

```
public void WyborMiejsca_Przyciski(object sender, RoutedEventArgs e)
{
   Button clickedButton = sender as Button;

   if (clickedButton == null) return;

   SolidColorBrush buttonBackground = clickedButton.Background as SolidColorBrush;

   if (buttonBackground == null) return;

   if (buttonBackground.Color == Colors.DarkGray)
   {
      clickedButton.Background = new SolidColorBrush(Colors.Green);
      Licznik++;
      Zliczanie.Text = Licznik.ToString();
      WybraneMiejsca.Add(clickedButton);
      Sala?.ListaPrzyciskow.Add(clickedButton.Name);
   }
   else if (buttonBackground.Color == Colors.Green)
   {
      clickedButton.Background = new SolidColorBrush(Colors.DarkGray);
      Licznik--;
      Zliczanie.Text = Licznik.ToString();
      WybraneMiejsca.Remove(clickedButton);
      Sala?.ListaPrzyciskow.Remove(clickedButton.Name);
   }
}
```

- FinalizacjaRezerwacji – Strona pozwalająca wybrać rodzaj i liczbę biletów oraz uzupełnić dane klienta.

- Konstruktor klasy, przyjmujący liczbę wybranych miejsc i listę przycisków reprezentujących te miejsca

```
public FinalizacjaRezerwacji(int licznik, List<Button> wybraneMiejsca)
{
  // Inicjalizacja komponentów interfejsu użytkownika
  InitializeComponent();

  // Dodanie możliwej liczby biletów do ComboBoxów
  for (int i = 0; i <= licznik; i++)
  {
    LiczbaBiletowNormalnychComboBox.Items.Add(i);
    LiczbaBiletowUlgowychComboBox.Items.Add(i);
  }

  // Sortowanie listy przycisków reprezentujących miejsca
  wybraneMiejsca = wybraneMiejsca.OrderBy(btn => Convert.ToInt32(btn.Content)).ToList();
  this.wybraneMiejsca = wybraneMiejsca;

  // Ustawienie tekstu informującego o wybranych miejscach
  TxtBoxWybraneMiejsca.Text = InformacjeOMiejscach();

  // Dodanie obsługi zdarzeń dla zmiany tekstu w TextBoxach
  ImieTextBox.TextChanged += TextBox_TextChanged;
  NazwiskoTextBox.TextChanged += TextBox_TextChanged;
  TelefonTextBox.TextChanged += TextBox_TextChanged;
  MailTextBox.TextChanged += TextBox_TextChanged;

  // Ustawienie domyślnych wartości ComboBoxów
  LiczbaBiletowNormalnychComboBox.SelectedIndex = wybraneMiejsca.Count;
  LiczbaBiletowUlgowychComboBox.SelectedIndex = 0;
```

```
    // Aktualizacja ceny
    AktualizujCene();
}
```

- Metoda generująca tekst informujący o wybranych miejscach

```
public string InformacjeOMiejscach()
{
    StringBuilder informacje = new StringBuilder();

    foreach (Button miejsce in wybraneMiejsca)
    {
        informacje.Append($"{miejsce.Content} ");
    }

    return informacje.ToString();
}
```

- Obsługa zmiany wybranej liczby biletów normalnych

```
private void LiczbaBiletowNormalnychComboBox_SelectionChanged(object sender,
SelectionChangedEventArgs e)
{
    liczbaBiletowNormalnych = LiczbaBiletowNormalnychComboBox.SelectedIndex;
    int dostepnaLiczbaMiejsc = wybraneMiejsca.Count;
    int liczbaBiletowUlgowych = dostepnaLiczbaMiejsc - liczbaBiletowNormalnych;

    // Ustawienie odpowiedniej wartości w ComboBoxie biletów ulgowych
    LiczbaBiletowUlgowychComboBox.SelectedIndex = liczbaBiletowUlgowych;

    // Aktualizacja ceny
    AktualizujCene();
}
```

- Obsługa zmiany wybranej liczby biletów ulgowych

```
private void LiczbaBiletowUlgowychComboBox_SelectionChanged(object sender,
SelectionChangedEventArgs e)
{
    liczbaBiletowUlgowych = LiczbaBiletowUlgowychComboBox.SelectedIndex;
    int dostepnaLiczbaMiejsc = wybraneMiejsca.Count;
    int liczbaBiletowNormalnych = dostepnaLiczbaMiejsc - liczbaBiletowUlgowych;

    // Ustawienie odpowiedniej wartości w ComboBoxie biletów normalnych
    LiczbaBiletowNormalnychComboBox.SelectedIndex = liczbaBiletowNormalnych;

    // Aktualizacja ceny
    AktualizujCene();
}
```

- Obsługa zmiany tekstu w TextBoxie

```
public void TextBox_TextChanged(object sender, TextChangedEventArgs e)
{
    TextBox textBox = (TextBox)sender;

    // Sprawdzenie, który TextBox został zmieniony
    if (textBox == ImieTextBox)
    {
        imieChanged = true;
    }
    else if (textBox == NazwiskoTextBox)
    {
        nazwiskoChanged = true;
    }
    else if (textBox == TelefonTextBox)
    {
```

```
            telefonChanged = true;
        }
        else if (textBox == MailTextBox)
        {
            mailChanged = true;
        }
        AktualizujCene();
    }
```

- Metoda aktualizująca cenę biletów

```
public decimal AktualizujCene()
    {
        int liczbaBiletow = liczbaBiletowNormalnych + liczbaBiletowUlgowych;
        decimal cena = (liczbaBiletowNormalnych * 20m) + (liczbaBiletowUlgowych * 16m);
        CenaTextBlock.Text = $"{cena} zł";
        return cena;
    }
```

- Obsługa przycisku finalizacji rezerwacji

```
    private void FinalizujRezerwacje_Click(object sender, RoutedEventArgs e)
    {
        // Wyświetlenie potwierdzenia rezerwacji
        MessageBoxResult result = MessageBox.Show("Czy na pewno chcesz dokonać rezerwacji?", "Potwierdzenie rezerwacji", MessageBoxButton.YesNo);
        if (result == MessageBoxResult.Yes)
        {
            // Sprawdzenie, czy dane klienta zostały uzupełnione
            if (!imieChanged || !nazwiskoChanged || !telefonChanged || !mailChanged)
            {
                MessageBox.Show("Proszę uzupełnić wszystkie dane klienta.", "Błąd", MessageBoxButton.OK, MessageBoxImage.Error);
                return;
            }
```

```
            // Pobranie danych klienta
            string imie = ImieTextBox.Text;
            string nazwisko = NazwiskoTextBox.Text;
            string telefon = TelefonTextBox.Text;
            string mail = MailTextBox.Text;

            try
            {
                // Utworzenie obiektu klienta
                klient = new Klient(imie, nazwisko, telefon, mail);

                // Wyświetlenie informacji o zakończonej rezerwacji
                MessageBox.Show("Rezerwacja zakończona pomyślnie!\nDane klienta:\n" +
klient.Imie + " " + klient.Nazwisko +
                    "\nTelefon: " + klient.Telefon + "\nEmail: " + klient.Mail);

                // Przejście do informacji o rezerwacji
                RezerwacjaNowa nowaRezerwacja = new RezerwacjaNowa(Sala,
InformacjeOMiejscach(), AktualizujCene(), klient, false, liczbaBiletowNormalnych,
liczbaBiletowUlgowych);
                InformacjeORezerwacji informacjeORezerwacji = new
InformacjeORezerwacji(nowaRezerwacja);
                this.NavigationService.Navigate(informacjeORezerwacji);
            }
            catch (Exception ex)
            {
                // Obsługa błędu podczas tworzenia obiektu klienta
                MessageBox.Show($"Błąd: {ex.Message}", "Błąd", MessageBoxButton.OK,
MessageBoxImage.Error);
            }
        }
        else
        {
```

```
            // Powrót do głównego okna w przypadku anulowania rezerwacji
            MainWindow mainWindow = new MainWindow();
            Application.Current.MainWindow = mainWindow;
            mainWindow.Show();
            Window.GetWindow(this).Close();
        }
    }
```

- InformacjeORezerwacji – Strona, na której wyświetlają się szczegóły rezerwacji.

Funkcje:

- Konstruktor klasy, przyjmujący rezerwacje jako parametr

```
    public InformacjeORezerwacji(RezerwacjaNowa rezerwacja)
    {
        InitializeComponent();
        // Ścieżka do pliku GIF
        string gifPath =
"Images/HNWT6DgoBc14riaEeLCzGYopkqYBKxpGKqfNWfgr368M9WNvwCmz6pCNJBicw2DEv2
YZDok3rZmom6Q9HZUAcqPCX2FEXqNsb85KMxpv67SxvvGS85BfSR6P2FP.gif";

        // Tworzenie obiektu Uri dla pliku GIF
        Uri gifUri = new Uri(gifPath, UriKind.RelativeOrAbsolute);

        // Ustawienie źródła animowanego obrazu
        ImageBehavior.SetAnimatedSource(GifImage, new BitmapImage(gifUri));

        // Pobranie danych z obiektu rezerwacji i utworzenie informacji do wyświetlenia
        int liczbaBiletowNormalnych = rezerwacja.LiczbaBiletowNormalnych;
        int liczbaBiletowUlgowych = rezerwacja.LiczbaBiletowUlgowych;
        string informacje = $"Dane klienta:\n{rezerwacja.Klient.Imie}
{rezerwacja.Klient.Nazwisko}\n" +
                            $"Telefon: {rezerwacja.Klient.Telefon}\nEmail: {rezerwacja.Klient.Mail}\n" +
                            $"Numer rezerwacji: {rezerwacja.NumerRezerwacji}\n" +
```

```
                $"Liczba biletów normalnych: {liczbaBiletowNormalnych}\n" +
                $"Liczba biletów ulgowych: {liczbaBiletowUlgowych}\n" +
                $"Cena: {rezerwacja.Cena} zł\n" +
                $"Numery miejsc: {rezerwacja.ZarezerwowaneMiejsca}";

    // Ustawienie tekstu w bloku tekstowym
    InformacjeTextBlock.Text = informacje;
}
```

- Obsługa zdarzenia kliknięcia przycisku Koniec

```
private void KoniecBtn_Click(object sender, RoutedEventArgs e)
{
    // Zamknięcie aplikacji
    Application.Current.Shutdown();
}

// Obsługa zdarzenia kliknięcia przycisku Menu
private void MenuBtn_Click(object sender, RoutedEventArgs e)
{
    // Przejście do głównego menu (MainWindow)
    MainWindow mainWindow = new MainWindow();
    Application.Current.MainWindow = mainWindow;
    mainWindow.Show();
    Window.GetWindow(this).Close();
}
```