|  |  |
| --- | --- |
| **Politechnika Świętokrzyska**  Wydział Elektrotechniki, Automatyki i Informatyki | |
| **Modelowanie i analiza systemów informatycznych** – Projekt | |
| **Temat:**  Zmiana unitermu poziomej operacji sekwencjonowania unitermów na pionową operację sekwencjonowania unitermów | **Opracowanie:**  Michał Kaczor (91268) 1IZ22B  **Data:**  01.01.2025 |

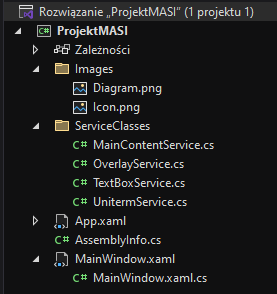
Spis treści

[1. Wstęp 3](#_Toc186646079)

[2. Wnioski 3](#_Toc186646080)

# Wstęp

# Najważniejsze części kodu



|  |
| --- |
| public partial class MainWindow : Window  {  public static bool isSwapped = false; // zmienna globalna określająca czy aktualnie unitermy są zamienione miejscami  private OverlayService overlayService;  private TextBoxService textBoxService;  private UnitermService unitermService;  private MainContentService mainContentService;  private TextBox[] textFields;  public MainWindow()  {  overlayService = new OverlayService();  textBoxService = new TextBoxService();  unitermService = new UnitermService();  mainContentService = new MainContentService();  InitializeComponent();  // Przypisujemy obsługę zdarzenia ValueChanged do suwaka ScaleSlider po inicjalizacji komponentów okna  ScaleSlider.ValueChanged += ScaleSlider\_ValueChanged;  textFields = new TextBox[] { HUValue1TextField, HUValue2TextField, VUValue1TextField, VUValue2TextField };  }  ... |

|  |
| --- |
| // Metoda zamieniająca miejscami unitermy  private void SwapUniterms(object sender, RoutedEventArgs e)  {  unitermService.Swap(sender, e, LeftRadioButton, RightRadioButton, HUValue1TextField, HUValue2TextField, VerticalUniterm, HorizontalUniterm, SwapButton, UndoButton, ClearFieldsButton, textFields);  } |

|  |
| --- |
| // Metoda zamieniająca miejscami unitermy  public void Swap(object sender, RoutedEventArgs e, RadioButton leftRadioButton, RadioButton rightRadioButton, TextBox hUValue1TextField, TextBox hUValue2TextField, StackPanel verticalUniterm, StackPanel horizontalUniterm,Button swapButton, Button undoButton, Button clearFieldsButton, TextBox[] textFields)  {  TextBoxService textBoxService = new TextBoxService();  // Sprawdzenie, czy wszystkie pola tekstowe są wypełnione  if (!textBoxService.IsNotEmpty(textFields))  {  MessageBox.Show("Wszystkie pola tekstowe muszą być wypełnione przed wykonaniem zamiany!", "Błąd", MessageBoxButton.OK, MessageBoxImage.Warning);  return;  }  bool isLeftSelected = leftRadioButton.IsChecked ?? false;  // Określenie elementów źródłowych i docelowych  var sourcePanel = isLeftSelected ? hUValue1TextField : hUValue2TextField;  var targetPanel = verticalUniterm;  var sourceParent = (Panel)sourcePanel.Parent;  var targetParent = (Panel)targetPanel.Parent;  // Przeniesienie elementów  sourceParent.Children.Remove(sourcePanel);  targetParent.Children.Remove(targetPanel);  sourceParent.Children.Add(targetPanel);  targetParent.Children.Add(sourcePanel);  // Rzutowanie sender na Button, ponieważ sender to obiekt, który wywołał metodę  Button? clickedButton = sender as Button;  // Sprawdzamy, czy rzutowanie powiodło się  if (clickedButton != null)  {  // Uzyskujemy nazwę przycisku  string buttonName = clickedButton.Name;  // Ustawiamy odpowiednie opcje dla pozostałych elementów ekranu w zależności od tego, któy przycisk był kliknięty  if (buttonName == "SwapButton")  {  MainWindow.isSwapped = true;  swapButton.IsEnabled = false;  undoButton.IsEnabled = true;  leftRadioButton.IsEnabled = false;  rightRadioButton.IsEnabled = false;  clearFieldsButton.IsEnabled = false;  sourcePanel.Visibility = Visibility.Hidden;  textBoxService.ToggleTextFields(textFields, false);  }  else if (buttonName == "UndoButton" || buttonName == "ResetButton")  {  MainWindow.isSwapped = false;  swapButton.IsEnabled = true;  undoButton.IsEnabled = false;  leftRadioButton.IsEnabled = true;  rightRadioButton.IsEnabled = true;  clearFieldsButton.IsEnabled = true;  sourcePanel.Visibility = Visibility.Visible;  textBoxService.ToggleTextFields(textFields, true);  }  }  } |

|  |
| --- |
| // Metoda obsługująca dopasowanie wielkości lini unitermu poziomego w zależności od zmiany wielkości panelu tekstowego  public void ScaleHUTextPanel(object sender, SizeChangedEventArgs e, StackPanel hUTextPanel, Path topPath)  {  if (hUTextPanel != null && topPath != null)  {  // Ustawienie szerokości TopPath na szerokość HUTextPanel  topPath.Width = hUTextPanel.ActualWidth;  // Dostosowanie danych geometrycznych Path  topPath.Data = new PathGeometry(new PathFigureCollection  {  new PathFigure  {  StartPoint = new Point(0, 0),  Segments = new PathSegmentCollection  {  new QuadraticBezierSegment  {  Point1 = new Point(topPath.Width / 2, -topPath.Height / 2),  Point2 = new Point(topPath.Width, 0)  }  }  }  });  }  } |

|  |
| --- |
| // Metoda obsługująca zmianę szerokości pola tekstowego w zależności od długości wprowadzonego tekstu  public void ScaleTextBox(object sender, TextChangedEventArgs e)  {  if (sender is TextBox textBox)  {  // Pobranie minimalnej szerokości z kontrolki TextBox  double minWidth = textBox.MinWidth;  // Mierzenie szerokości tekstu w TextBox  var formattedText = new FormattedText(  textBox.Text,  System.Globalization.CultureInfo.CurrentCulture,  FlowDirection.LeftToRight,  new Typeface(textBox.FontFamily, textBox.FontStyle, textBox.FontWeight, textBox.FontStretch),  textBox.FontSize,  Brushes.Black,  new NumberSubstitution(),  1);  // Obliczanie nowej szerokości na podstawie tekstu  double newWidth = formattedText.Width + 10; // Dodajemy 10 dla lepszego odstępu  // Ustawianie szerokości TextBox z zachowaniem minimalnej szerokości  textBox.Width = Math.Max(newWidth, minWidth);  }  } |

# Diagram przypadków użycia

# Diagram klas

# Diagram aktywności

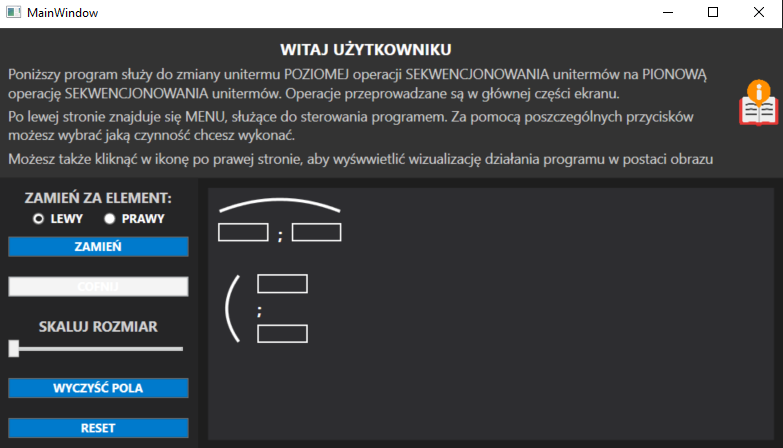
# Diagram sekwencji

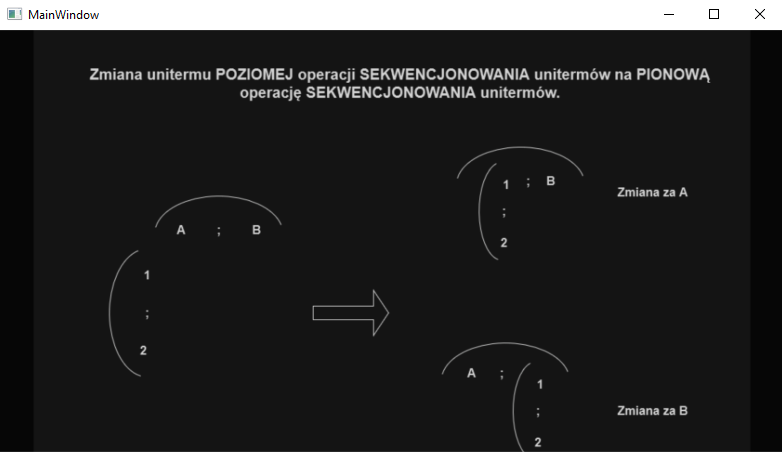
# Diagram warstw

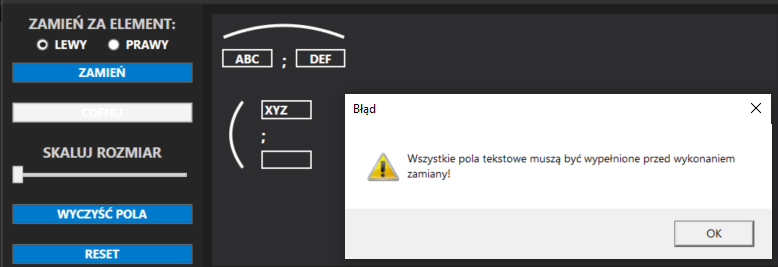
# Diagram komponentów

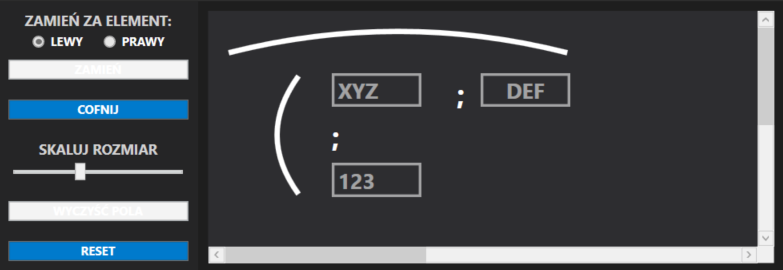
# Struktura bazy danych

# Zdjęcia z działania aplikacji

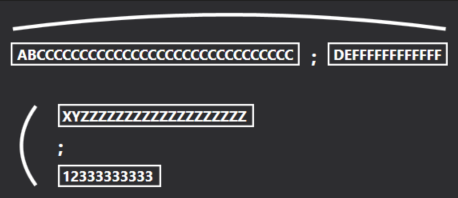












# Wnioski