Menu, karty i skróty — edytor notatek

Czas: 2 × 45 min | praca indywidualna | Wykład 2.08

Cel: Zbudujesz edytor notatek z menu, paskiem narzędzi, paskiem stanu i kartami TabControl, w którym menu, przyciski i skróty klawiszowe wywołują te same komendy.

Po zajęciach potrafisz:

- buduje Menu, ToolBar i StatusBar z wiązaniami do komend
- wyświetla kolekcję jako karty TabControl
- dodaje działające skróty klawiszowe KeyBinding
- wykorzystuje komendy wbudowane ApplicationCommands

Sprzęt i materiały

- komputer z Windows 11 i dostępem do internetu
- pendrive lub konto w repozytorium do zapisania projektu

Oprogramowanie

- **Visual Studio Community 2026 (obciążenie „Programowanie aplikacji klasycznych platformy .NET")** — darmowe dla uczniów i nauczycieli; w pracowni zainstalowane visualstudio.microsoft.com/pl/downloads
- **.NET 10 SDK (LTS)** — instalowany razem z Visual Studio; sprawdzenie: `dotnet --version` dotnet.microsoft.com/pl-pl/download/dotnet/10.0
- **Pakiet NuGet CommunityToolkit.Mvvm 8.4** — instalacja: Zarządzaj pakietami NuGet nuget.org/packages/CommunityToolkit.Mvvm

Bezpieczeństwo

- Zapisuj projekt w swoim folderze (np. `C:\Users\<login>\source\repos`), a na koniec zajęć skopiuj go na pendrive lub wyślij do repozytorium.
- Nie uruchamiaj programów i pakietów z nieznanych źródeł; pakiety NuGet instaluj tylko te, które wskazuje zadanie.
- Rób przerwy dla oczu (zasada 20-20-20) — praca z projektantem okien wymaga długiego patrzenia w ekran.

Przebieg zadania

1 ViewModele

Projekt WPF `Notatki` z CommunityToolkit.Mvvm.

1. Klasa statyczna `Tekst.LiczSlowa(string?)` (bez WPF).
2. `NotatkaViewModel`: Tytul, Tresc, Zmieniona; właściwości LiczbaZnakow, LiczbaSlow, Naglowek (tytuł z gwiazdką przy zmianach).
3. `EdytorViewModel`: kolekcja notatek, `Aktywna`, `PokazPasek`, `Komunikat`; komendy Nowa, Zamknij, Zapisz (aktywne, gdy jest karta).

2 Okno

Interfejs aplikacji biurowej.

```
<Window.InputBindings>
  <KeyBinding Key="N" Modifiers="Control" Command="{Binding NowaCommand}" />
  <KeyBinding Key="S" Modifiers="Control" Command="{Binding ZapiszCommand}" />
  <KeyBinding Key="W" Modifiers="Control" Command="{Binding ZamknijCommand}" />
</Window.InputBindings>
```

1. Menu Plik (Nowa, Zapisz, Zamknij kartę), Edycja (Wytnij, Kopiuj, Wklej — `ApplicationCommands`), Widok (Pasek stanu — IsCheckable).
2. ToolBar z ikonami z czcionki Segoe Fluent Icons (zapasowo Segoe MDL2 Assets), ToolTip i AutomationProperties.Name.
3. StatusBar: znaki, słowa, komunikat; ukrywany przez BooleanToVisibilityConverter.
4. TabControl: `ItemTemplate` (nagłówek) i `ContentTemplate` (TextBox treści).

3 Skróty i sprawdzenie

Spójność poleceń.

1. Dodaj KeyBinding dla Ctrl+N, Ctrl+S, Ctrl+W; w menu tylko `InputGestureText`.
2. Sprawdź każde polecenie trzema drogami (menu, pasek, klawiatura).
3. Zamknij wszystkie karty — przyciski Zapisz i Zamknij muszą się wyszarzyć.
4. Zatwierdź pracę w Git.

W sprawozdaniu umieść

- ☐ kod ViewModeli
- ☐ XAML okna
- ☐ tabela poleceń (menu / pasek / skrót)
- ☐ zrzut okna z kilkoma kartami

Pytania kontrolne

1. Dlaczego InputGestureText nie uruchamia skrótu?
2. Czym różni się ItemTemplate od ContentTemplate w TabControl?
3. Dlaczego Kopiuj i Wklej działają bez własnego kodu?
4. Jak sprawić, by pasek stanu dało się ukryć?

Kryteria oceny

Kryterium	Punkty
ViewModele i licznik słów	5
Menu, ToolBar, StatusBar	5
TabControl z kolekcji	4
Skróty i CanExecute	3
Pytania	3
Razem	**20**

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).