Programowanie wizualne

opracował: Paweł Wojciechowski 2023/24

Lab .NET MAUI 1

Zagadnienia do opanowania:

- Aplikacja .NET MAUI
- Pojemniki (Layouts)
- Podstawowe wiązanie danych i proste zdarzenia

Temat: Aplikacja .NET MAUI

1. Utworzyć nowy projekt Aplikacja platformy .NET MAUI. Przejrzeć zawartość utworzonych plików i katalogów. Zwrócić uwagę na pliki: MauiProgram.cs, App.xaml i App.xaml.cs, AppShell.xaml.cs i AppShell.xaml, MainPage.xaml i MainPage.xaml.cs, oraz pliki Colors.xaml i Styles.xaml (te dwa ostatnie znajdują się w katalogu Resources/Styles).

Zadania i pytania kontrolne:

1.1. które metody/linie kodu po kolei decydują o wyświetleniu w aplikacji zawartości MainPage.xaml?
1.2. Czy można utworzyć bezpośrednio MainPage bez wykorzystania AppShell? Jak?
1.3. Proszę zmienić kolor przycisków na czerwony.
1.4. Jak aplikacja reaguje na zmianę rozmiaru okna? Zmodyfikować Button na głównej stronie tak, żeby nie rozciągał się na szerokość okna. Które właściwości którego komponentu odpowiadają za to, że button nie jest tak szeroki jak okno (zostaje na początku i końcu spory kawałek bez względu na rozmiar okna.
1.5. Jak zrobić, żeby samochodzik był większy – tzn. jego rozmiar rósł bez ograniczeń razem z rozmiarem okna?
1.6. Dlaczego, dla pewnego rozmiaru okna przycisk jest praktycznie przycięty od dołu i wciąż nie pojawia się pasek przewijania? Jak to zmienić?
1.7. Jak wymusić żeby pojawił się również drugi pasek przewijania (poziomy)?

2. Dodać do projektu nową stronę (Dodaj nowy element→ .NET MAUI ContentPage (XAML)) o nazwie LayoutPage. Dodać tę stronę do AppShell.xaml. (Żeby ta strona była widoczna, to albo musi być przed opisana w znaczniku ShellContent przed stroną Home albo należy obie wrzucić do środka znacznika FlyoutItem). Ta strona służyć będzie do testowania poszczególnych rozkładów (layouts). Proszę wrzucić do pojemnika kilka komponentów np. Button o różnej zawartości (dla Buttona jest to właściwość Text). Można również użyć komponentu BoxView (tutaj uwaga na parametry, które decydują o rozmiarze WidthRequest i HeightRequest no i Color).
   2.1. VerticalStackLayout:
      2.1.1. przetestować działanie właściwości Margin, Padding i Spacing.
      2.1.2. Co się dzieje z komponentami, gdy ustawimy im właściwości WidthRequest, MinimumWidthRequest i MaximumWidthRequest? Jak komponenty z ustawionymi powyższymi właściwościami reagują na zmianę rozmiaru okna? Co z właściwościami Height? Co się stanie, jeśli ustawimy WidthRequest > MinimumWidthRequest? Czy można ustawić Maximum mniejsze niż Minimum? Czy kolejność definiowania właściwości ma znaczenie?
      2.1.3. Ile miejsca zajmuje cały pojemnik? Proszę to sprawdzić wrzucając pojemnik VerticalStackLayout do środka pojemnika ScrollView. Dodatkowo można ustawić kolor tła (właściwość Background) dla pojemnika. Co się stanie gdy ustawimy dodatkowo HeightRequest albo MaximumHeightRequest?
   2.2. HorizontalStackLayout:
      2.2.1. Sprawdzić zachowanie komponentów przy powyższych ustawieniach dla pojemnika HorizontalStackLayout. Które parametry decydują w tym przypadku o rozmiarze pojemnika i komponentów?
   2.3. FlexLayout:
      2.3.1. Wrzucić do pojemnika 10 przycisków. Ustawić wartość właściwości Direction na Rows oraz Wrap na Wrap. Sprawdzić jak właściwości VerticalOptions ustawione na wartość inną niż Fill oraz HeightRequest wpływa na zachowanie pojemnika. Proszę pamiętać, że cokolwiek będzie widać w momencie zmiany wielkości okna. Sprawdzić działanie (czy sposób rozmieszczenia komponentów w zależności od wartości parametru Direction komponentu FlexLayout.
   2.4. Grid:
      2.4.1. Utworzyć pojemnik Grid z dwoma kolumnami i trzema wierszami. Rozmieścić w nim po jednym buttonie w każdej z komórek (właściwości Grid.Row i Grid.Column).
         2.4.1.1. Jak wpływają wartości parametrów HorizontalOptions,VerticalOptions,

HeightRequest i WidthRequest na rozmiar buttonów?

2.4.1.2. Co się stanie jeśli w jednej komórce dodatkowo dołożymy drugi komponent? Proszę dodać do wybranej komórki komponent BoxView z ustawionymi parametrami WidthRequest i HeightRequest.

2.4.1.3. Co się stanie, jeśli do jednej komórki wstawimy np. VerticalStackLayout z np. 3 przyciskami?

2.4.1.4. Usunąć jeden z przycisków i korzystając z właściwości Grid.RowSpan umieścić jeden z buttonów w dwóch komórkach.

2.5. AbsoluteLayout – utworzyć pojemnik tego typu, a następnie korzystając z wartości proporcjonalnych i bezwzględnych umieścić w każdym rogu kwadrat o rozmiarze 20x20, oraz niebieski krzyż przecinający na połowę komponent w pionie i poziomie.

3. Wiązanie danych.

Utworzyć nową stronę o nazwie BindingPage, dodać ją do AppShella, a następnie zaimplementować aplikację, która:

3.1. wyświetla jakiś napis jako etykietę (Label)
3.2. ma komponent Slider, którego wartości są np. z przedziału od 5 do 50
3.3. ma komponent Entry, który wyświetla aktualną wartość slidera
3.4. Rozmiar czcionki z Ektykiety jest powiązany z wartością właściwości Value slidera.
3.5. Dodatkowo dodać do strony AbsoluteLayout o szerokości np. 50 jednostek i kolorowym tle (właściwość Background). Na środku przy lewej krawędzi proszę wyświetlić czerwony komponent BoxView o rozmiarze 20x20.
3.6. Dodać kolejny Slider, który przy zmianie ma generować zdarzenie (ValueChanged). W metodzie obsługi tego zdarzenia ustawić pozycję czerwonego kwadrata tak, aby był on nad znacznikiem slidera

Podpowiedzi do ostatniego podpunktu:

- ustawić wartość min dla slidera na 0, a maksymalną na 1,
- zmieniać wartość proporcjonalną X dla kwadratu (w kodzie) na taką samą jak wartość slidera.
- Nową wartość można odczytać z argumentów metody obsługi zdarzenia
- ustawienie wartości AbsoluteLayout dla BoxView w kodzie to:
  `AbsoluteLayout.SetLayoutBounds, gdzie pierwszym argumentem jest nazwa komponentu BoxView, a drugim struktura Rect z 4 parametrami takimi jak AbsouluteLayout.LayoutBounds.`