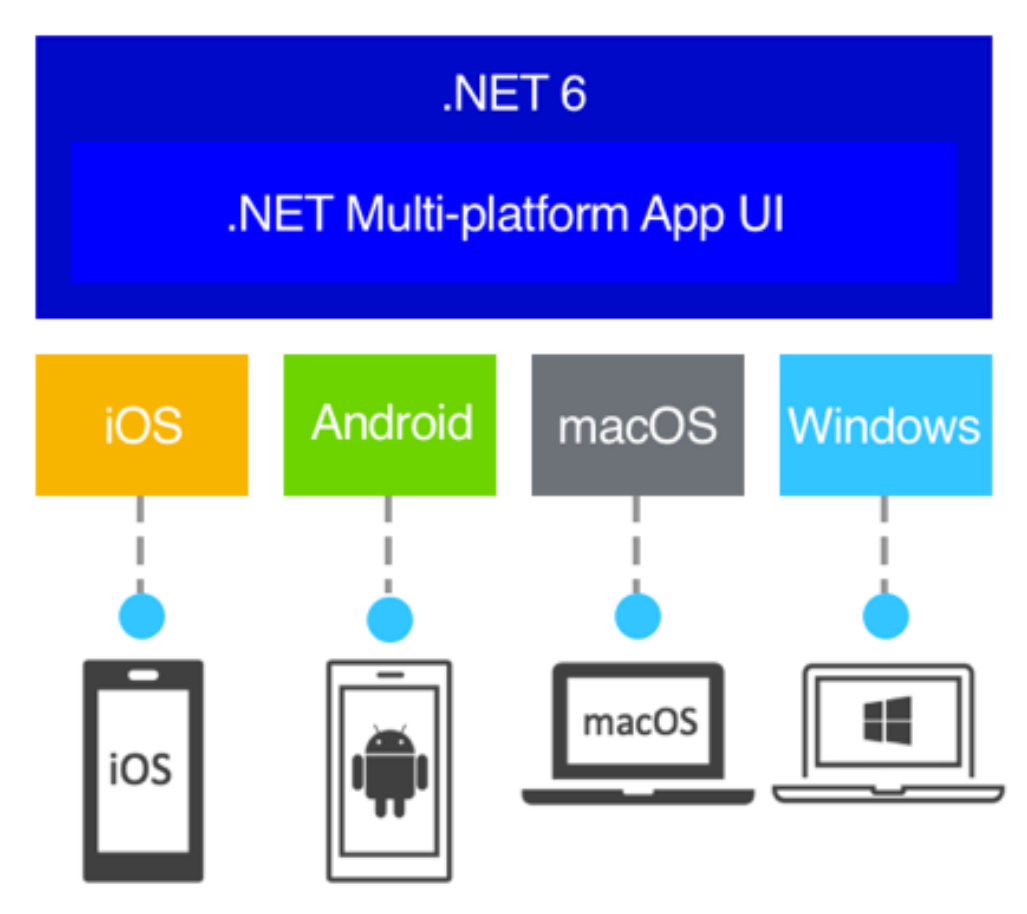
Curs 7

Aplicatii multi-platforma cu .NET MAUI

Ce este .NET MAUI?

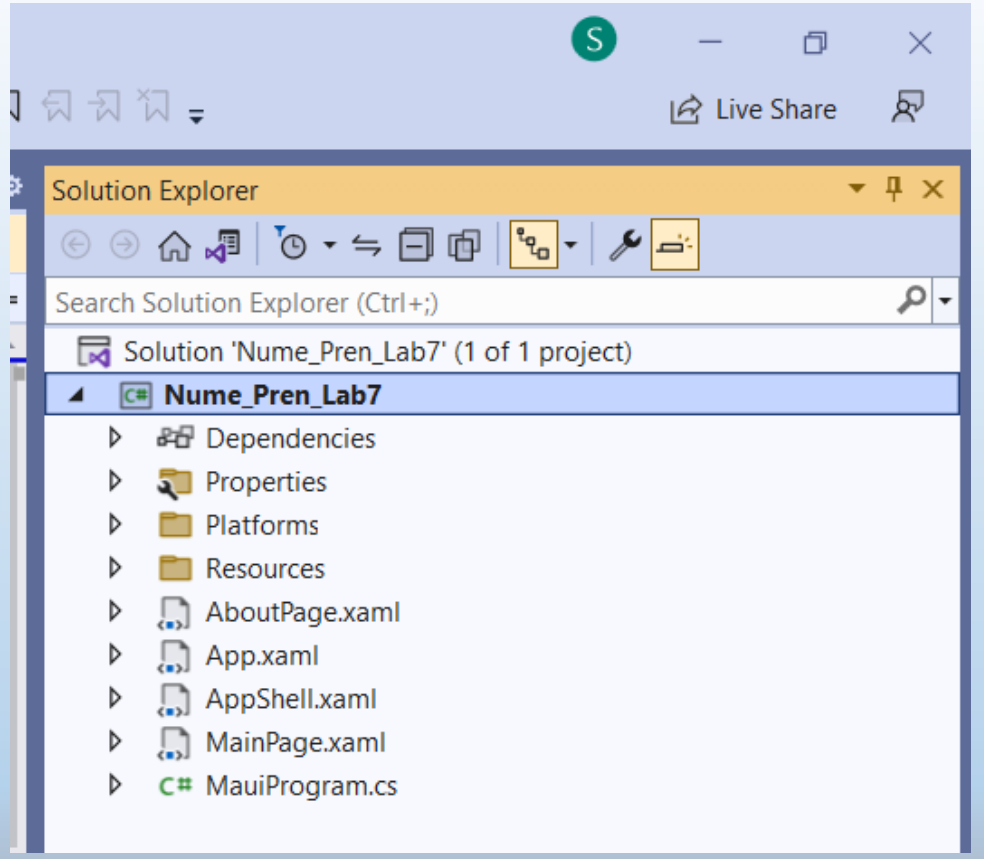
- Este un framework multi-platforma pentru crearea de aplicații mobile și desktop cu C# și XAML.
- Putem dezvolta aplicații care pot rula pe Android, iOS, macOS și Windows dintr-o singură bază de cod partajată.
- Reprezinta evolutia lui Xamarin.Forms (fiind extins de la aplicatii mobile la scenarii de tip desktop)



Structura unei aplicatii .NET MAUI

- *MauiProgram.cs*

- fișierul care configureaza si pornește aplicația.
- punct de intrare al aplicației
- indica clasa App definită de fișierul App.xaml.



Structura unei aplicatii .NET MAUI

- *App.xaml* si *App.xaml.cs*
  - Fișierul .xaml conține marcaj XAML, iar fișierul de cod conține cod pentru a interacționa cu marcajul XAML.
  - Fișierul App.xaml conține resurse XAML la nivel de aplicație, cum ar fi culori, stiluri sau șabloane.
  - Fișierul App.xaml.cs conține în cod care instanțiază aplicația Shell
- *AppShell.xaml și AppShell.xaml.cs*
  - Acest fișiere definesc ierarhia vizuală a aplicației.
- MainPage.xaml și MainPage.xaml.cs
  - Aceasta este pagina de pornire afișată de aplicație. Fișierul MainPage.xaml definește UI (interfața de utilizator) a paginii. MainPage.xaml.cs conține codul din spatele XAML

XAML

- XAML (eXtensible Application Markup Language) este un limbaj bazat pe XML permite organizarea obiectelor în ierarhii părinte-copil.
- XAML permite dezvoltatorilor să definească UI în aplicațiile .NET MAUI utilizând markup in loc de cod.
- Avantaje:
  - XAML este adesea succint
  - Ierarhia părinte-copil inerentă în XML permite XAML să imite cu o mai mare claritate vizuală ierarhia părinte-copil a obiectelor interfeței cu utilizatorul.
- Dezavantaje:
  - Gestiunea evenimentelor trebuie sa fie realizata intr-un fisier separat cu cod sursa
  - Nu exista designer visual pentru elementele XAML. Toate elementele de interfata trebuie editate manual (putem folosi insa hot reload pentru a vizualiza UI pe masura ce il editam)

Anatomia unui fisier XAML

- <ContentPage> este obiectul rădăcină pentru clasa AboutPage. ContentPage poate avea un singur obiect copil.
- <VerticalStackLayout> este singurul obiect copil al ContentPage. Tipul VerticalStackLayout poate avea mai mulți copii. Acest control aranjează copiii pe verticală, unul după altul.
- <Image> afișează o imagine, în acest caz folosește imaginea dotnet_bot.png care vine cu fiecare proiect .NET MAUI.
- <Label> afișeaza text
- <Button> declaseaza evenimentul Clicked. Putem rula cod ca răspuns la evenimentul Clicked Clicked="OnCounterClicked"
- Evenimentul Clicked al butonului este atribuit handler-ului de evenimente OnCounterClicked, definit în fișierul code-behind.

Sintaxa XAML

- XAML este conceput în principal pentru instanțierea și inițializarea obiectelor
- Label este un *object element* obiect .NET MAUI reprezentat ca un element XML.
- Text, VerticalOptions, FontAttributes and FontSize sunt *property attributes* – proprietati reprezentate ca atribute XML
- In cel de al doilea exemplu TextColor a devenit *property element*

```
<Label Text="Hello, XAML!"
       VerticalOptions="Center"
       FontAttributes="Bold"
       FontSize="18"
       TextColor="Aqua" />
```

```
<Label Text="Hello, XAML!"
VerticalOptions="Center"
FontAttributes="Bold" FontSize="18">
 <Label.TextColor>
Aqua
 </Label.TextColor>
 </Label>
```

Proprietati semantice

- Folosite pentru a defini informații despre controale
- Clasa SemanticProperties definește următoarele proprietăți atașate:
  - Description (tip string)- reprezintă o descriere
  - Hint (tip Sir) care este similar cu Description, dar oferă context suplimentar, cum ar fi scopul unui control
  - HeadingLevel care permite ca un element să fie marcat ca titlu pentru a organiza interfața de utilizare și a facilita navigarea

Attached Properties

```
<Grid>
<Grid.RowDefinitions>
<RowDefinition Height="Auto" />
 <RowDefinition Height= "100" />
</Grid.RowDefinitions>
 <Grid.ColumnDefinitions>
<ColumnDefinition Width="Auto" />
<ColumnDefinition Width="100" />
</Grid.ColumnDefinitions>
<Label Text=“My Text"
Grid.Row="1" Grid.Column="1"
TextColor="Purple">
</Grid>
```

- Sunt definite de clasa Grid, dar sunt setate la nivelul copiilor Gridului

Content properties

```
<ContentPage
    xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="XamlSamples.XamlPlusCodePage"
    Title="XAML + Code Page">
<ContentPage.Content>
 <Grid> ... </Grid>
</ContentPage.Content>
</ContentPage>
```

-nu sunt sunt obligatorii pentru ca toatele elementele .NET MAUI pot o avea un singur element Content

XAML Markup extensions

- permit setarea proprietăților la obiecte sau valori care sunt referite indirect din alte surse (ex. de obicei, utilizam XAML pentru a seta proprietățile unui obiect la valori explicite, cum ar fi un string, un număr, etc.)
- sunt deosebit de importante pentru partajarea obiectelor și constantele de referință utilizate în cadrul unei aplicații
- markup extensions XAML sunt denumite astfel deoarece sunt susținute de cod în clasele care implementează IMarkupExtension. Putem sa definim markup extensions custom

Resurse comune

```
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
x:Class="XamlSamples.SharedResourcesPage" Title="Shared Resources Page">
    <StackLayout>
    <Button Text="Do this!" HorizontalOptions="Center" VerticalOptions="Center"
    BorderWidth="3" Rotation="-15" TextColor="Red" FontSize="24" />
    <Button Text="Do that!" HorizontalOptions="Center" VerticalOptions="Center"
    BorderWidth="3" Rotation="-15" TextColor="Red" FontSize="24" />
    </StackLayout>
</ContentPage>
```

Dacă una dintre aceste proprietăți trebuie schimbată, preferam să facem modificarea o singură dată.

Dictionar de resurse

- un dicționar cu perechi de tip chei string și valori asociate

```
<ContentPage
xmlns="http://schemas.microsoft.com/dotnet/2021/
maui"
xmlns:x="http://schemas.microsoft.com/winfx/2009/
xaml" x:Class="XamlSamples.SharedResourcesPage"
Title="Shared Resources Page">

<ContentPage.Resources>

<LayoutOptions x:Key="horzOptions"
Alignment="Center" />

<LayoutOptions x:Key="vertOptions"
Alignment="Center" /> </ContentPage.Resources>

...

</ContentPage>
```

```
<Button Text="Do this!"
HorizontalOptions="{StaticResource horzOptions}“
 VerticalOptions="{StaticResource vertOptions}“
 BorderWidth="3“
 Rotation="-15"
TextColor="Red"
FontSize="24" />
```

StaticResource este întotdeauna delimitată cu {} și include cheia din dicționarul de resurse

DynamicResource - pentru cheile de dicționar asociate cu valori care s-ar putea modifica în timpul execuției, în timp ce StaticResource accesează elementele din dicționar o singură dată când elementele de pe pagină sunt construite

Pagini Shell in .NET MAUI

- Un obiect ShellContent reprezintă obiectul ContentPage pentru fiecare FlyoutItem sau Tab
- Paginile sunt create de obicei la cerere, ca răspuns la navigare
- Acest lucru se realizează prin utilizarea extensiei de markup *DataTemplate* pentru a seta proprietatea *ContentTemplate* a fiecărui obiect ShellContent la un obiect ContentPage

```
<Shell …>

    <TabBar>
        <ShellContent
            Title="Welcome"
            ContentTemplate="{DataTemplate
local:MainPage}"
             />

        <ShellContent
            Title="About"
            ContentTemplate="{DataTemplate
local:AboutPage}"
             />

        <ShellContent
            Title="My Shopping Lists"
            ContentTemplate="{DataTemplate
local:ListEntryPage}"
             />
    </TabBar>

</Shell>
```

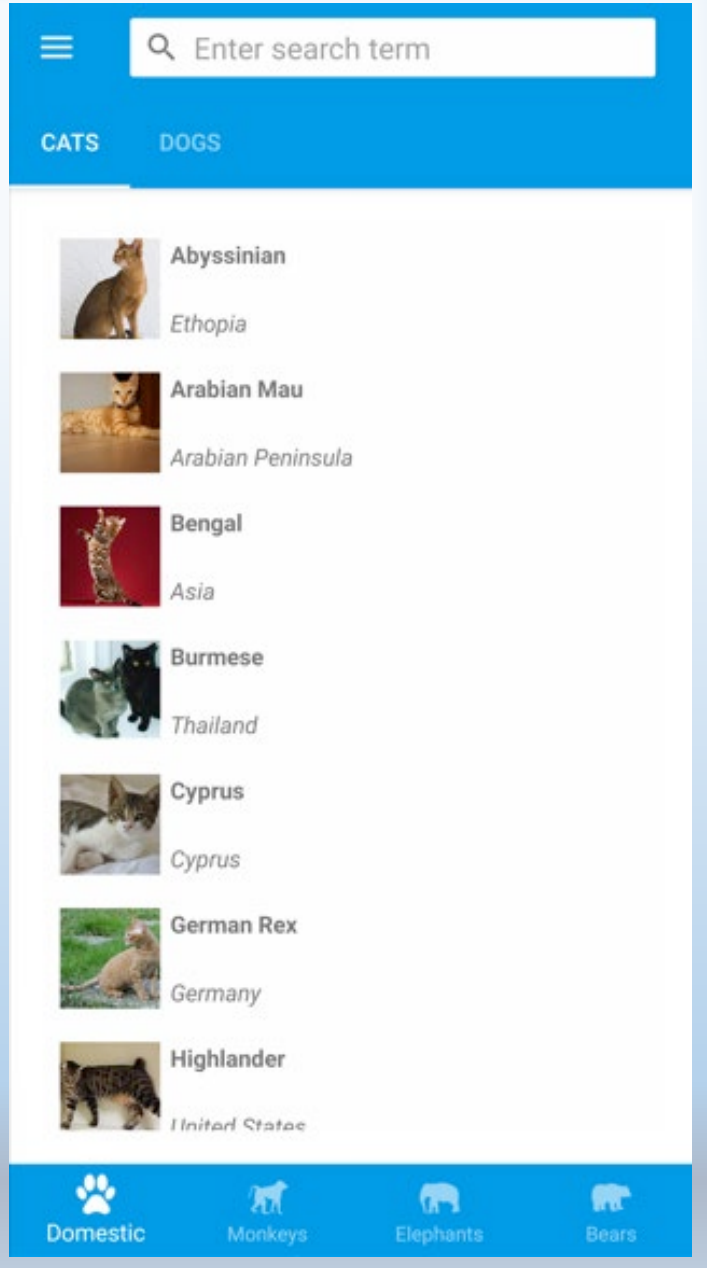
Descrierea ierarhiei vizuale a unei aplicatii

- Ierarhia vizuală a unei aplicații .NET MAUI Shell este descrisă în clasa Shell, pe care șablonul de proiect o numește AppShell.
- O clasă Shell subclasată poate contine din trei obiecte ierarhice principale:
  - FlyoutItem sau TabBar. Un FlyoutItem reprezintă unul sau mai multe elemente de tip flyout. Un TabBar reprezintă bara de file și ar trebui utilizată atunci când modelul de navigare pentru aplicație începe cu file și nu necesită un instrument derulant. Fiecare obiect FlyoutItem sau obiect TabBar este un copil al obiectului Shell.
  - Tab, care reprezintă conținut grupat, navigabil. Fiecare obiect Tab este un copil al unui obiect FlyoutItem.
  - ShellContent, care reprezintă obiectele ContentPage pentru fiecare filă. Fiecare obiect ShellContent este un copil al unui obiect Tab. Atunci când într-o filă sunt prezente mai multe obiecte ShellContent, obiectele vor fi navigabile prin filele de sus.

Descrierea ierarhiei - exemplu

- <Shell xmlns="http://schemas.microsoft.com/dotnet/2021/maui" xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" xmlns:views="clr-namespace:Xaminals.Views" x:Class="Xaminals.AppShell"> ...
- <FlyoutItem FlyoutDisplayOptions="AsMultipleItems">
  - <Tab Title="Domestic" Icon="paw.png">
    - <ShellContent Title="Cats" Icon="cat.png" ContentTemplate="{DataTemplate views:CatsPage}" /> <ShellContent Title="Dogs" Icon="dog.png" ContentTemplate="{DataTemplate views:DogsPage}" />
  - </Tab>
  - <ShellContent Title="Monkeys" Icon="monkey.png" ContentTemplate="{DataTemplate views:MonkeysPage}" /> <ShellContent Title="Elephants" Icon="elephant.png" ContentTemplate="{DataTemplate views:ElephantsPage}" /> <ShellContent Title="Bears" Icon="bear.png" ContentTemplate="{DataTemplate views:BearsPage}" />
- </FlyoutItem>
- ... </Shell>

- Aceste obiecte nu reprezintă nicio interfață cu utilizatorul, ci mai degrabă organizarea ierarhiei vizuale a aplicației. Shell va prelua aceste obiecte și va produce interfața de navigare cu utilizatorul pentru conținut.



Navigarea in .NET MAUI

- navigare bazată pe URI, care utilizează rute pentru a naviga la orice pagină din aplicație

- navigare bazata pe ierarhie

Navigarea bazata pe URI

- Navigarea se realizează prin invocarea metodei GoToAsync, din clasa Shell
- Când navigarea este pe cale să fie efectuată, evenimentul *Navigating* este declanșat și evenimentul *Navigated* este declanșat când navigarea se termină.
- Navigarea cu ruta absoluta
  - `await Shell.Current.GoToAsync("//animals/monkeys");`
- Navigarea cu ruta relativa
  - `await Shell.Current.GoToAsync("monkeydetails");`

  - animals
    - domestic
      - cats
      - dogs
    - monkeys
    - elephants
    - bears
  - about

Rute in .NET MAUI

- URI-urile de navigare pot avea trei componente:
  - O rută, care definește calea către conținut care există ca parte a ierarhiei vizuale Shell.
  - O pagina. Paginile care nu există în ierarhia vizuală Shell pot fi introduse în stiva de navigare de oriunde dintr-o aplicație Shell. De exemplu, o pagină de detalii nu va fi definită în ierarhia vizuală Shell, dar poate fi introdusă în stiva de navigare după cum este necesar.
  - Unul sau mai mulți parametri de interogare. Parametrii de interogare sunt parametri care pot fi transferați către pagina de destinație în timpul navigării.
- Când un URI de navigare include toate cele trei componente, structura este: //route/page?queryParameters

Inregistrarea rutelor

- <Shell ...>
  - <FlyoutItem ... Route="animals">
    - <Tab ... Route="domestic">
      - <ShellContent ... Route="cats" />
      - <ShellContent ... Route="dogs" />
    - </Tab>
    - <ShellContent ... Route="monkeys" />
    - <ShellContent ... Route="elephants" />
    - <ShellContent ... Route="bears" />
  - </FlyoutItem>
  - <ShellContent ... Route="about" />
  - </Shell>

- Rutele pot fi definite pentru obiectele FlyoutItem, TabBar, Tab și ShellContent
- Se seteaza prin proprietatea Route
  - animals
    - domestic
      - cats
      - dogs
    - monkeys
    - elephants
    - bears
  - about
- Ruta absoluta pentru dogs: //animals/domestic/dogs
- ArgumentException va fi aruncata la pornirea aplicației dacă este detectată o rută duplicată

Inregistrarea rutelor paginilor detaliu

- Paginile de tip detaliu nu se afla in ierarhia vizuala a clasei Shell
- În constructorul clasei Shell, rute suplimentare pot fi înregistrate în mod explicit pentru orice pagini de detalii care nu sunt reprezentate în ierarhia vizuală Shell.
- Acest lucru se realizează cu metoda Routing.RegisterRoute:

```
public AppShell()
{
InitializeComponent();
Routing.RegisterRoute("monkeydetails", typeof(MonkeyDetailPage));
Routing.RegisterRoute("beardetails", typeof(BearDetailPage));
Routing.RegisterRoute("catdetails", typeof(CatDetailPage));
Routing.RegisterRoute("dogdetails", typeof(DogDetailPage));
Routing.RegisterRoute("elephantdetails", typeof(ElephantDetailPage));
}
```

- Aceste pagini de detalii pot fi apoi navigate folosind navigarea bazată pe URI, de oriunde în cadrul aplicației
- Rutele pentru astfel de pagini sunt cunoscute ca rute globale

Rute Relative

- Navigarea poate fi efectuată și prin specificarea unui URI relativ, valid ca argument pentru metoda GoToAsync.
- Sistemul de rutare va încerca să potrivească URI-ul cu un obiect ShellContent.
- Prin urmare, dacă toate rutele dintr-o aplicație sunt unice, navigarea poate fi efectuată prin specificarea numelui unic al rutei ca URI relativ. Sunt acceptate următoarele formate de rută relative:
  - route Ierarhia rutei va fi căutată pentru ruta specificată, în sus de la poziţia curentă. Pagina potrivită va fi adaugata în stiva de navigare.
  - /route Ierarhia rutei va fi căutată de pe ruta specificată, în jos de la poziția curentă. Pagina potrivită va fi adaugata în stiva de navigare.
  - //route Ierarhia rutei va fi căutată pentru ruta specificată, în sus de la poziția curentă. Pagina potrivită va înlocui stiva de navigare.
  - ///route Ierarhia rutei va fi căutată pentru ruta specificată, în jos de la poziţia curentă. Pagina potrivită va înlocui stiva de navigare.

Navigarea ierharhica

- Pagini
  - Modale
  - NeModale
- In designul UI “modal” se refera necesitatea ca utilizatorul sa interactioneze cu aplicatia inainte ca aplicatia sa continue
- Cand o fereastra modala este afisata utilizatorului, utilizatorul nu poate sa se intoarca la ferestra principala, ci trebuie sa intractioneze cu fereastra modala mai intai
- In general utilizam ferestre modale cand aplicatia are nevoie de informatii de la utilizator si nu dorim ca utilizatorul sa se intoarca la pagina anterioara decat dupa ce furnizeaza informatiile
- O fereastra care nu este modala se numeste fereastra nemodala

Navigarea ierarhica

- Mecanismul de navigare intre pagini implementeaza pagini modale si nemodale prin definirea a doua metode care pot fi apelate de o pagina pentru a naviga catre alta pagina
  - `Task PushAsync(Page page)` – navigheaza catre o pagina nemodala
  - `Task PushModalAsync(Page page)` - navigheaza catre o pagina modala
- Pentru a naviga catre pagina anterioara sunt definite metodele:
  - `Task<Page> PopAsync()`
  - `Task<Page> PopModalAsync()`

Navigarea catre o alta pagina

- Utilizam proprietatea Navigation a obiectului pagina, codul pentru a naviga catre alta pagina va arata asa:

```
await Navigation.PushAsync(new MyNewPage());
await Navigation.PushModalAsync(new MyNewModalPage());
```