**ЮГОЗАПАДЕН УНИВЕРСИТЕТ "НЕОФИТ РИЛСКИ"**

ПРИРОДО-МАТЕМАТИЧЕСКИ ФАКУЛТЕТ

**КАТЕДРА "ИНФОРМАТИКА"**

КУРСОВ ПРОЕКТ

по

Приложения за мобилни устройства

на тема:   
РАЗРАБОТВАНЕ НА ПРИЛОЖЕНИЕ ЗА „ПТИЦИ В БЪЛГАРИЯ“

**Изготвили: Проверил:**

20251421001 Цветелин Георгиев доц. д-р Радослава Кралева

спец. ИСТ 4

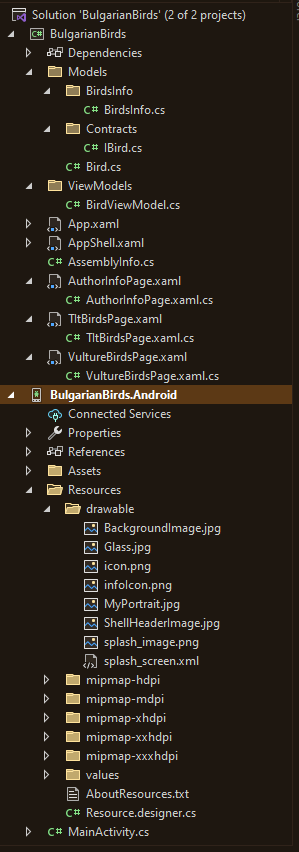
Благоевград

2023

# 1. Информация на приложението

Приложението е разработено с цел да предостави кратко описание и снимки за птици в България, както и да се освободя от изпит по дисциплината. Има две основни страници, като едната предоставя възможност да се избира визуализиране на информация за видове синигери, а другата за видове лешояди. Има и трета страница за автора с малко текст и снимка. И трите се достъпват чрез страничното **Flyout** меню.

Визуален изглед на дървото на проекта (Solution Explorer):



# 2. Сплаш екран

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <resources>  <style name="MainTheme" parent="MainTheme.Base">  <!-- As of Xamarin.Forms 4.6 the theme…-->  </style>  <style name="MyTheme.Splash" parent="MainTheme">  <item name="android:windowBackground">@drawable/splash\_screen</item>  </style>  </resources> |

styles.xml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <resources>  <color name="launcher\_background">#FFFFFF</color>  <color name="colorPrimary">#3F51B5</color>  <color name="colorPrimaryDark">#303F9F</color>  <color name="colorAccent">#FF4081</color>  <color name="SplashScreenColor">#FFFFFF</color>  </resources> |

colors.xml

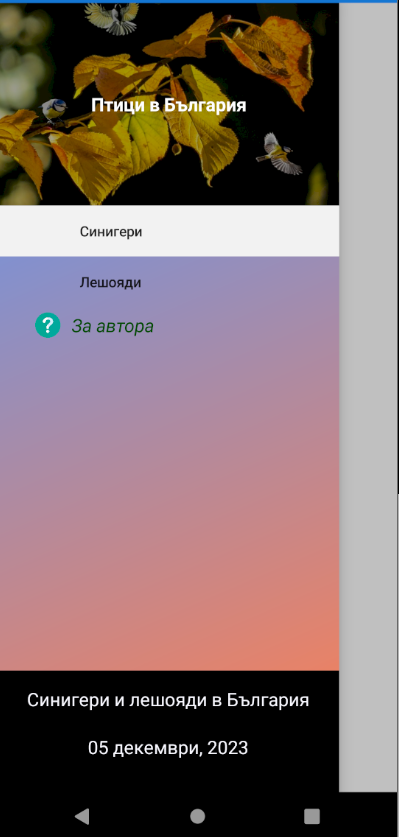
|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <layer-list xmlns:android="http://schemas.android.com/apk/res/android">  <item>  <color android:color="@color/SplashScreenColor"/>  </item>  <item>  <bitmap android:src="@drawable/splash\_image"  android:tileMode="disabled"  android:gravity="center"/>  </item>  </layer-list> |

splash\_screen.xml

MainActivity.cs

|  |
| --- |
| using Android.App;  using Android.Content.PM;  using Android.OS;  using Android.Runtime;  namespace App\_9\_2.Droid  {  [Activity(Label = "BulgarianBirds", Icon = "@drawable/icon", Theme = "@style/MyTheme.Splash", MainLauncher = true,  ConfigurationChanges = ConfigChanges.ScreenSize | ConfigChanges.Orientation | ConfigChanges.UiMode  | ConfigChanges.ScreenLayout | ConfigChanges.SmallestScreenSize)]  public class MainActivity : global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity  {  protected override void OnCreate(Bundle savedInstanceState)  {  base.SetTheme(Resource.Style.MainTheme);  base.OnCreate(savedInstanceState);  Xamarin.Essentials.Platform.Init(this, savedInstanceState);  global::Xamarin.Forms.Forms.Init(this, savedInstanceState);  LoadApplication(new App());  }  public override void OnRequestPermissionsResult(int requestCode, string[] permissions, [GeneratedEnum] Android.Content.PM.Permission[] grantResults)  {  Xamarin.Essentials.Platform.OnRequestPermissionsResult(requestCode, permissions, grantResults);  base.OnRequestPermissionsResult(requestCode, permissions, grantResults);  }  }  } |

# 3. Flyout меню



Двете страници за птиците са описани във визуалната йерархия на приложението чрез елемента **ShellContet**. Страницата за автора е **MenuItem**, което не я включва във визуалната йерархия през xaml кодът, а в конструктора на съпровождащия c# файл („**Routing.RegisterRoute();**”). Това създава така нареченият път (**route)**, което позволява страницата да бъде достъпвана от всяко място в приложението чрез **URI**-базирана навигация.

* Изгледът на съответния елемент се дефинира чрез тага **Shell.MenuItemTemplate.** Навигацията в този случай се извършва чрез имплементация на интерфейса **ICommand**, който се имплементира от функцията **HelpCommand**, на която се подава функцията за пренасочване. То се извършва чрез асинхронната функция: “**await Shell.Current.GoToAsync(nameof(AuthorInfoPage));”**.
* При реализирана навигация чрез **Shell** всяка страница бива зареждана на момента на отваряне, което пести ресурси. За разлика от това, при **Tabbed Page** всички страници се зареждат наведнъж, което може да доведе до забавяне, особено ако страницата е зададена като коренна (**root**).
* Фонът е оцветен с **LinearGradientBrush**, намиращ се в **Shell.FlyoutBackground**. Ефектът на хедърът, описан в **Shell.FlyoutHeaderTemplate**, се състои от изображение и черен **BoxView** върху него с намалена прозрачност, за да се постигне затъмняващият ефект. Футърът, дефиниран чрез **Shell.FlyoutFooter**, показва текущата дата, взета от **local** именуваното пространство, което достъпва папката **ViewModels**. От нея се достига статичното пропърти **CurrentDate** във файла **BirdViewModel**, което взима текущата дата и я форматира.

AppShell.xaml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <Shell xmlns="http://xamarin.com/schemas/2014/forms"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:views="clr-namespace:App\_9\_2" x:DataType="views:AppShell"  xmlns:local="clr-namespace:App\_9\_2.ViewModels"  x:Class="App\_9\_2.AppShell"  FlyoutBackdrop="Silver"  BackgroundColor="Green">  <Shell.FlyoutHeaderTemplate>  <DataTemplate>  <Grid BackgroundColor="Black"  HeightRequest="200">  <Image Aspect="AspectFill"  Source="@drawable/ShellHeaderImage.jpg"  Opacity="0.65" />  <Label Text="Птици в България"  TextColor="White"  FontAttributes="Bold"  FontSize="Medium"  HorizontalTextAlignment="Center"  VerticalTextAlignment="Center" />  </Grid>  </DataTemplate>  </Shell.FlyoutHeaderTemplate>  <Shell.MenuItemTemplate>  <DataTemplate>  <Grid ColumnDefinitions="0.2\*, 0.8\*" RowDefinitions="35">  <Image Grid.Column="0"  Source="@drawable/infoIcon.png"  HorizontalOptions="EndAndExpand"  Margin="5"/>  <Label Grid.Column="1"  TextColor="#004b00"  Text="За автора"  FontSize="Medium"  FontAttributes="Italic"  VerticalTextAlignment="Center"  HorizontalOptions="StartAndExpand"/>  </Grid>  </DataTemplate>  </Shell.MenuItemTemplate>  <Shell.FlyoutBackground>  <LinearGradientBrush StartPoint="0,0"  EndPoint="1,1">  <GradientStop Color="CornflowerBlue"  Offset="0.1" />  <GradientStop Color="Coral"  Offset="1.0" />  </LinearGradientBrush>  </Shell.FlyoutBackground>  <ShellContent  Title="Синигери"  ContentTemplate="{DataTemplate views:TitBirdsPage}"/>  <ShellContent  Title="Лешояди"  ContentTemplate="{DataTemplate views:VultureBirdsPage}"/>  <MenuItem  Command="{Binding HelpCommand}"  CommandParameter="https://learn.microsoft.com/xamarin/xamarin-forms/app-fundamentals/shell" />  <Shell.FlyoutFooter>  <Grid RowDefinitions="0.5\*, 0.5\*" BackgroundColor="Black" HeightRequest="120">  <Label Grid.Row="0" Text="Синигери и лешояди в България"  TextColor="GhostWhite" FontSize="Medium"  VerticalOptions="CenterAndExpand"  HorizontalOptions="Center"  HorizontalTextAlignment="Center"/>  <Label Grid.Row="1" Text="{Binding Source={x:Static local:BirdViewModel.CurrentDate}}"  TextColor="GhostWhite" FontSize="Medium"  VerticalOptions="StartAndExpand"  HorizontalOptions="Center"  HorizontalTextAlignment="Center"/>  </Grid>  </Shell.FlyoutFooter>  </Shell> |

AppShell.xaml.cs

|  |
| --- |
| using System.Windows.Input;  using Xamarin.Forms;  namespace App\_9\_2  {  public partial class AppShell : Shell  {  private Command helpCommand;  public AppShell()  {  InitializeComponent();  BindingContext = this;  Routing.RegisterRoute(nameof(AuthorInfoPage), typeof(AuthorInfoPage));  }  public string AuthorInfoText => "За автора";  public ICommand HelpCommand  {  get  {  if (helpCommand == null)  {  helpCommand = new Command(Help);  }  return helpCommand;  }  }  private async void Help()  {  await Shell.Current.GoToAsync(nameof(AuthorInfoPage));  Shell.Current.FlyoutIsPresented = false;  }  }  } |

# 4. Основни страници

И двете основни страници за птиците са абсолютно еднакви, без заглавието и името на файла. В конструктора на c# файла на всяка страница се зарежда списък със съответните птици (синигери или лешояди). Понеже не успях да намеря начин това да се контролира чрез навигацията, има две почти идентични страници. Цялото текстово и мултимедийно съдържание на страницата се извлича от **BindingContext** на страницата, който е свързан с **BirdViewModel.cs** файлът. Целял съм да постигна MVVM архитектура чрез описаните файлове.

* Най-отгоре на страницата има **Picker** елемент от който се избира видът птица.
* Основният изглед се състои от **Frame** елементи с **Label**-и в тях, а в средата **CarouselView** с изображения. Местата на обитаване и описанието на птиците са също и в **ScrollView**, за да се постигне еднаквост на интерфейса при различни големини на дисплея.
* Фонът е постигнат отново чрез затъмняване на изображение с **BoxView** с намалена прозрачност. За тази цел има **Grid**, който обгръща основното съдържание на страницата, а над него са двата елемента съставящи фона.
* Белите правоъгълници са множество **BoxView** елементи поставени под елементите съдържащи снимката, като те заедно са обградени от **Grid**, за да не се изменя стойността на изместване с всеки нов елемент, за разлика от **StackView** конструкция.
* Цялото съдържание на страницата е в **Grid** със скрит черен фон с намалена прозрачност и текущата снимка на цял екран. Тези елементи се визуализират когато се натисне върху настоящата снимка на основния изглед и служи за мащабиране. Мащабирането става чрез жестът **Tap** два пъти върху вече показаната снимка на цял екран и може да се повтори няколко пъти. При единично кликване се връща в оригинален цялостен размер. При тапване на снимката докато е в оригинален цялостен размер тя изчезва и се връща към основния изглед на страницата. Същото може да се постигне и чрез плъзгане на долу (**Swipe Gesture**).

TitBirdsPage.xaml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  x:Class="App\_9\_2.TitBirdsPage"  Title="Tits page">  <Grid BackgroundColor="#e5fbe5">  <StackLayout>  <Picker x:Name="PickerTitBird" HorizontalOptions="Center"  Title="Изберете вид синигер" Margin="0, 10, 0, 0"  TitleColor="Black" FontAttributes="Italic,Bold"  HorizontalTextAlignment="Center"  WidthRequest="280" HeightRequest="45"  ItemsSource="{Binding Birds}"  ItemDisplayBinding="{Binding Name}"  SelectedItem="{Binding SelectedBird}">  </Picker>  <Grid VerticalOptions="FillAndExpand">  <BoxView VerticalOptions="FillAndExpand" BackgroundColor="Black"/>  <Image Source="@drawable/BackgroundImage"  Aspect="AspectFill" Opacity="0.6"/>  <StackLayout Padding="20" HorizontalOptions="Fill">  <Frame BackgroundColor="DarkGreen" CornerRadius="25"  Margin="20, -10, 20, 0" Padding="10">  <Label Text="{Binding SelectedBird.Name}"  HorizontalOptions="Center" TextColor="White" FontSize="Large"/>  </Frame>  <Frame BackgroundColor="White" CornerRadius="25"  Margin="70, 0, 70, 0" Padding="7">  <Label Text="{Binding SelectedBird.EndangeredInfo}"  HorizontalOptions="Center" TextColor="{Binding SelectedBird.IsEndangeredLabelColor}"  FontSize="Medium" FontAttributes="Italic,Bold"/>  </Frame>  <Frame Margin="5" Padding="5" CornerRadius="15"  BackgroundColor="#d8ffd8" HeightRequest="75">  <ScrollView VerticalOptions="FillAndExpand" >  <Label Text="{Binding SelectedBird.ResidesInAreas}"  TextColor="Black" Margin="15, 5, 5, 5" FontAttributes="Italic"  FontSize="Medium" VerticalOptions="FillAndExpand"/>  </ScrollView>  </Frame>  <Grid Padding="0, 7, 0, 0" RowDefinitions="Auto, \*">  <Frame Grid.Row="0" BackgroundColor="Black" CornerRadius="30"  HasShadow="True" MinimumHeightRequest="192" HeightRequest="192"  HorizontalOptions="FillAndExpand" VerticalOptions="CenterAndExpand">  <CarouselView x:Name="CarouselElement" ItemsSource="{Binding SelectedBird.ImageUrls}"  PeekAreaInsets="20" Margin="0, 10, 0, 10">  <CarouselView.ItemsLayout>  <LinearItemsLayout ItemSpacing="10" SnapPointsAlignment="Center"  SnapPointsType="MandatorySingle" Orientation="Horizontal"/>  </CarouselView.ItemsLayout>  <CarouselView.ItemTemplate>  <DataTemplate>  <Image Source="{Binding}" Aspect="AspectFill">  <Image.GestureRecognizers>  <TapGestureRecognizer Tapped="TapGestureRecognizer\_Tapped\_PreviewImage"/>  </Image.GestureRecognizers>  </Image>  </DataTemplate>  </CarouselView.ItemTemplate>  </CarouselView>  </Frame>  <BoxView BackgroundColor="LightGray" HorizontalOptions="Center" VerticalOptions="Center"  WidthRequest="15" HeightRequest="20"  TranslationX="-120" TranslationY="-100"/>  <BoxView BackgroundColor="LightGray" HorizontalOptions="Center" VerticalOptions="Center"  WidthRequest="15" HeightRequest="20"  TranslationX="-70" TranslationY="-100"/>  <BoxView BackgroundColor="LightGray" HorizontalOptions="Center" VerticalOptions="Center"  WidthRequest="15" HeightRequest="20"  TranslationX="-20" TranslationY="-100"/>  <BoxView BackgroundColor="LightGray" HorizontalOptions="Center" VerticalOptions="Center"  WidthRequest="15" HeightRequest="20"  TranslationX="30" TranslationY="-100"/>  <BoxView BackgroundColor="LightGray" HorizontalOptions="Center" VerticalOptions="Center"  WidthRequest="15" HeightRequest="20"  TranslationX="80" TranslationY="-100"/>  <BoxView BackgroundColor="LightGray" HorizontalOptions="Center" VerticalOptions="Center"  WidthRequest="15" HeightRequest="20"  TranslationX="130" TranslationY="-100"/>  <BoxView BackgroundColor="LightGray" HorizontalOptions="Center" VerticalOptions="Center"  WidthRequest="15" HeightRequest="20"  TranslationX="-120" TranslationY="101"/>  <BoxView BackgroundColor="LightGray" HorizontalOptions="Center" VerticalOptions="Center"  WidthRequest="15" HeightRequest="20"  TranslationX="-70" TranslationY="101"/>  <BoxView BackgroundColor="LightGray" HorizontalOptions="Center" VerticalOptions="Center"  WidthRequest="15" HeightRequest="20"  TranslationX="-20" TranslationY="101"/>  <BoxView BackgroundColor="LightGray" HorizontalOptions="Center" VerticalOptions="Center"  WidthRequest="15" HeightRequest="20"  TranslationX="30" TranslationY="101"/>  <BoxView BackgroundColor="LightGray" HorizontalOptions="Center" VerticalOptions="Center"  WidthRequest="15" HeightRequest="20"  TranslationX="80" TranslationY="101"/>  <BoxView BackgroundColor="LightGray" HorizontalOptions="Center" VerticalOptions="Center"  WidthRequest="15" HeightRequest="20"  TranslationX="130" TranslationY="101"/>  <Frame Grid.Row="1" Margin="10" Padding="5" CornerRadius="15"  VerticalOptions="StartAndExpand" BackgroundColor="#003b00">  <ScrollView VerticalOptions="FillAndExpand" >  <Label Text="{Binding SelectedBird.Description}"  TextColor="White" Margin="15, 5, 5, 5" FontAttributes="Italic"  FontSize="Medium" VerticalOptions="FillAndExpand"/>  </ScrollView>  </Frame>  </Grid>  </StackLayout>  </Grid>  </StackLayout>  <BoxView x:Name="ZoomedModeDarkening" VerticalOptions="FillAndExpand" HorizontalOptions="FillAndExpand"  BackgroundColor="Black" Opacity="0.5">  <BoxView.GestureRecognizers>  <TapGestureRecognizer Tapped="TapGestureRecognizer\_Tapped\_ZoomedImage"/>  </BoxView.GestureRecognizers>  </BoxView>  <Image x:Name="ZoomedImage" Aspect="AspectFit"  VerticalOptions="FillAndExpand" HorizontalOptions="FillAndExpand">  <Image.GestureRecognizers>  <TapGestureRecognizer Tapped="TapGestureRecognizer\_Tapped\_ZoomedImage\_Twice" NumberOfTapsRequired="2"/>  <TapGestureRecognizer Tapped="TapGestureRecognizer\_Tapped\_ZoomedImage"/>  <PanGestureRecognizer PanUpdated="PanGestureRecognizer\_PanUpdated"/>  </Image.GestureRecognizers>  </Image>  </Grid>  </ContentPage> |

4.1 Програмна част към основните страници

И двата файла отново са еднакви, с разликата на заредената информация. Декларирам няколко променливи нужни за изчисленията на мащабирането. В конструктора подготвям обектите на снимката за мащабиране и фона, създавам **ViewModel**-а и му подавам съответният списък със птици за страницата, след което го задавам като **BindingContext**. Добавил съм коментари към методите и се надявам да са достатъчни за целите на това описание. Готов съм да ги обсъждам по-нашироко при представянето на проекта.

TitBirdsPage.xaml.cs

|  |
| --- |
| using App\_9\_2.Models.BirdsInfo;  using App\_9\_2.Models.Contracts;  using App\_9\_2.ViewModels;  using System;  using System.Collections.Generic;  using Xamarin.Forms;  namespace App\_9\_2  {  public partial class TitBirdsPage : ContentPage  {  private const double ScaleWith = 0.3;  private int zoomLevel = 1;  private double currentScale = 1;  public TitBirdsPage()  {  InitializeComponent();  List<IBird> birds = BirdsInfo.GetTitBirds();  BindingContext = new BirdViewModel(birds);  ZoomedImage.IsEnabled = false;  ZoomedImage.IsVisible = false;  ZoomedModeDarkening.IsEnabled = false;  ZoomedModeDarkening.IsVisible = false;  }  /// <summary>  /// Pan ZoomedImage. If pan reaches a certain value of the Y axis,  /// set 'zoomLevel' to 1 and go to the method for hiding the ZoomedImage.  /// </summary>  private void PanGestureRecognizer\_PanUpdated(object sender, PanUpdatedEventArgs e)  {  ZoomedImage.TranslationX += e.TotalX;  ZoomedImage.TranslationY += e.TotalY;  if (ZoomedImage.TranslationY > 270 \* (zoomLevel \* 0.7))  {  zoomLevel = 1;  TapGestureRecognizer\_Tapped\_ZoomedImage(sender, e);  }  }  /// <summary>  /// Zoom the 'ZoomedImage' element by scaling it. New scale is calculated and applied.  /// Also increments 'zoomLevel' by 1.  /// </summary>  private void TapGestureRecognizer\_Tapped\_ZoomedImage\_Twice(object sender, EventArgs e)  {  if (zoomLevel <= 5)  {  currentScale += ScaleWith \* (zoomLevel \* 2);  ZoomedImage.ScaleTo(currentScale, 150, Easing.CubicIn);  zoomLevel++;  }  }  /// <summary>  /// Return 'ZoomedImage' to original scale if it's already zoomed.  /// Hide and disable it if it's not already zoomed.  /// </summary>  private void TapGestureRecognizer\_Tapped\_ZoomedImage(object sender, EventArgs e)  {  if (zoomLevel > 1)  {  ZoomedImage.ScaleTo(1, 150, Easing.CubicIn);  ZoomedImage.TranslateTo(0, 0, 150, Easing.Linear);  zoomLevel = 1;  }  else  {  ZoomedImage.TranslateTo(0, 0, 250);  ZoomedImage.ScaleTo(1, 250);  ZoomedImage.IsEnabled = false;  ZoomedImage.IsVisible = false;  ZoomedModeDarkening.IsEnabled = false;  ZoomedModeDarkening.IsVisible = false;  }  currentScale = 1;  }  /// <summary>  /// Show and enable 'ZoomedImage'.  /// </summary>  private void TapGestureRecognizer\_Tapped\_PreviewImage(object sender, EventArgs e)  {  ZoomedImage.Source = (sender as Image).Source;  ZoomedImage.IsEnabled = true;  ZoomedImage.IsVisible = true;  ZoomedModeDarkening.IsEnabled = true;  ZoomedModeDarkening.IsVisible = true;  }  }  } |

5. Моделът Bird

Основният и единствен модел на приложението е обектът **Bird** и той държи нужната информация за представяните птици. Придружава го и интерфейс, който не е нужен за текущата реализация на приложението. Реших да го добавя за упражнение и за да се придържам към **SOLID** принципите за чист код. Идеята е в бъдеще основната идея за формата на базовия обект птица да се съхранява в интерфейса и да може да се прави композиция с допълнителни функционалности при нужда. Например при добавяне на обект **EditableBird**, който представлява обект с възможността за редактиране и въвеждане на данни от потребител, ще се използва базовия **IBird** интерфейс, както и допълнителен **IEditable**.

* Пропъртито **EndangeredInfo** се променя според булевата стойност **IsEndangered.**
* Тук пазя и цветът на надписът **„Застрашен“ / „Не е застрашен“** от основните страници. Червен ако птицата е застрашена или зелен, ако не е.

IBird.cs

|  |
| --- |
| using System.Collections.Generic;  namespace App\_9\_2.Models.Contracts  {  public interface IBird  {  string Name { get; set; }  bool IsEndangered { get; set; }  string EndangeredInfo { get; }  string ResidesInAreas { get; set; }  List<string> ImageUrls { get; set; }  string Description { get; set; }  string IsEndangeredLabelColor { get; }  }  } |

Bird.cs

|  |
| --- |
| using App\_9\_2.Models.Contracts;  using System;  using System.Collections.Generic;  namespace App\_9\_2.Models  {  public class Bird : IBird  {  public string Name { get; set; }  public bool IsEndangered { get; set; }  public string EndangeredInfo  {  get => IsEndangered ? "Застрашен." : "Не е застрашен.";  }  public List<string> ImageUrls { get; set; } = new List<string>();  public string ResidesInAreas { get; set; }  public string Description { get; set; }  public string IsEndangeredLabelColor => IsEndangered ? "#b30000" : "DarkGreen";  }  } |

5. ViewModel **обектът BirdViewModel**

Този обект служи за моят **ViewModel** и в него се съхранява колекция от **IBird** обекти, в нашия случай синигери или лешояди. Колекцията се подава през конструктора, като за текуща птица се взима първия елемент от нея.

* Направил съм я да е от тип **IEnumerable**, за да не ограничавам бъдещи разработчици с определен тип колекция.
* Тук се изчислява и форматира текущата дата, нужна на футъра на **Flyout** менюто.
* Пропъртито **SelectedBird** е обвързано с **Binding**, следователно когато се промени чрез **Picker** обекта от основната страница се влиза в неговия **set** метод. Там се проверява дали е променена стойността и ако да, се задава новата стойност като текуща и се извиква метод, който на свой ред извиква стандартното вградено събитие на **Xamarin** – **PropertyChangedEventHandler**.

BirdViewModel.cs

|  |
| --- |
| using App\_9\_2.Models.Contracts;  using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Linq;  namespace App\_9\_2.ViewModels  {  public class BirdViewModel : INotifyPropertyChanged  {  private IBird \_selectedBird;  public BirdViewModel(IEnumerable<IBird> inputBirds)  {  Birds = inputBirds;  SelectedBird = Birds.First();  }  public IEnumerable<IBird> Birds { get; private set; }  public static string CurrentDate => DateTime.Now.ToString("dd MMMM, yyyy", new System.Globalization.CultureInfo("bg"));  public IBird SelectedBird  {  get => \_selectedBird;  set  {  if (value != \_selectedBird)  {  \_selectedBird = value;  OnPropertyChanged(nameof(SelectedBird));  }  }  }  public event PropertyChangedEventHandler PropertyChanged;  private void OnPropertyChanged(string propertyName)  => PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));  }  } |

5. Статичният клас BirdsInfo

Този статичен клас създава двете ни нужни колекции от птици, съответно синигери и лешояди с необходимата информация. Състои се от два статични метода, всеки от които създава нужната колекция с обекти с ръчно попълнени данни. Снимките са линкове към сайта на „Българско дружество за защита на птиците“ и/или „Уикипедия“.

BirdViewModel.cs

|  |
| --- |
| using App\_9\_2.Models.Contracts;  using System.Collections.Generic;  namespace App\_9\_2.Models.BirdsInfo  {  public static class BirdsInfo  {  public static List<IBird> GetTitBirds()  {  List<IBird> birds = new List<IBird>  {  new Bird()  {  Name = "Голям синигер",  IsEndangered = false,  ImageUrls = {"https://bspb.org/wp-content/uploads/2021/07/P\_major\_2\_S\_Spasov.jpg", "https://atlas.bspb.org/wp-content/uploads/sites/5/2021/09/1-12\_12-12-2020\_%D0%A0%D0%B0%D0%B4%D0%BE%D1%81%D0%BB%D0%B0%D0%B2%D0%A0%D0%B0%D0%BD%D0%BA%D0%BE%D0%B2\_Pmajor-credit.jpg" },  ResidesInAreas = "Среща се в цялата страна, с изключение на високопланинската безлесна зона.",  Description = "Дължина на тялото: 13 – 15 см\r\nНай-често срещаният синигер у нас. Главата е черна с бели бузи. Коремът е жълт, разделен от черна надлъжна ивица, която при мъжките е по-широка отколкото при женските. При младите тя почти липсва. Гърбът е жълто-зелен, а крилата имат син оттенък."  },  new Bird()  {  ………………..  };  return birds;  }  public static List<IBird> GetVultureBirds()  {  List<IBird> birds = new List<IBird>()  {  new Bird()  {  Name = "Египетски лешояд",  IsEndangered = true,  ImageUrls = { "https://atlas.bspb.org/wp-content/uploads/sites/5/2021/08/IMG\_7234\_July15\_S\_Spasov-credit.jpg", "https://atlas.bspb.org/wp-content/uploads/sites/5/2021/08/Dimitar\_Gradinarov\_Ptici\_At\_14-04-03\_0058-credit.jpg" },  ResidesInAreas = "Източните Родопи, Източна Стара планина, Провадийско–Роякското плато, Ломовете.",  Description = "Дължината на тялото е 60 – 70 (до 85) cm, а размахът на крилете – 150 – 170 cm. Теглото му достига 1,6 – 2,5 kg. Няма изразен полов диморфизъм. Оперението му е в светли кремави тонове, като краищата на крилете и всички махови пера са тъмни. Главата е с оголени участъци около човката в жълт цвят. Полетът е предимно планиращ."  },  new Bird()  {  ………………..  };  return birds;  }  }  } |

5. Страницата AuthorPage

Това е допълнителната страница с информация за мен. Има снимка, чрез **Frame** и **BoxView** елементи е направена да изглежда като закачена картина. До нея има малко информация за мен, а най-отдолу цитати в **ScrollView**.

* Фонът съм оцветил със **LinearGradientBrush**, а данните отдясно са ръчно попълнени **Label** елементи.
* Цитатите най-отдолу са в **ScrollView** съдържащ един **Label** елемент, който взима текста от придружаващия страницата c# файл. Така и не можах да направя височината адаптивна, като при описанието на основните страници. Като е по-малък екрана излиза извън границите му. Затова реших просто да му задам някаква статична височина, която да не влиза в конфликт с повечето дисплеи.

AuthorPage.xaml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  x:Class="App\_9\_2.AuthorInfoPage">  <StackLayout>  <StackLayout.Background>  <LinearGradientBrush StartPoint="0,0"  EndPoint="1,1">  <GradientStop Color="White"  Offset="0.0" />  <GradientStop Color="LightGreen"  Offset="0.2" />  <GradientStop Color="Green"  Offset="0.5" />  <GradientStop Color="DarkGreen"  Offset="1.0" />  </LinearGradientBrush>  </StackLayout.Background>  <Grid RowDefinitions="Auto, 30, \*, Auto">  <Grid Grid.Row="0" Grid.Column="0" ColumnDefinitions="Auto, \*">  <Grid Grid.Column="0" VerticalOptions="StartAndExpand" HorizontalOptions="StartAndExpand">  <BoxView WidthRequest="5" HeightRequest="100"  Color="Black" HorizontalOptions="Center" VerticalOptions="Center"  TranslationX="-45" TranslationY="-153" Rotation="60"/>  <BoxView WidthRequest="5" HeightRequest="100"  Color="Black" HorizontalOptions="Center" VerticalOptions="Center"  TranslationX="45" TranslationY="-153" Rotation="-60"/>  <BoxView WidthRequest="15" HeightRequest="15"  Color="Black" HorizontalOptions="Center" VerticalOptions="Center"  TranslationX="0" TranslationY="-175"/>  <Frame Background="Brown" HorizontalOptions="Start"  HeightRequest="280" MinimumHeightRequest="280"  Margin="20, 60, 20, 20" Padding="10">  <Frame Background="Black" Padding="5">  <Grid>  <Image Source="@drawable/MyPortrait" Aspect="AspectFill"/>  <Image Source="@drawable/Glass" WidthRequest="10"  Opacity="0.2" Aspect="AspectFill"/>  </Grid>  </Frame>  </Frame>  </Grid>  <StackLayout Grid.Column="1" Margin="0, 69, 0, 0" HorizontalOptions="CenterAndExpand">  <Label Text="Цветелин Георгиев" HorizontalTextAlignment="End" VerticalOptions="Center"  FontSize="Large" FontAttributes="Bold" TextColor="Black" Margin="0, 0, 20, 0"/>  <Label Text="Информационни Системи и технологии" HorizontalTextAlignment="End" VerticalOptions="Center"  FontSize="Subtitle" FontAttributes="Bold,Italic" TextColor="Black" Margin="0, 0, 20, 0"/>  <Label Text="20251421001" HorizontalTextAlignment="End" VerticalOptions="Center"  FontSize="Subtitle" FontAttributes="Bold,Italic" TextColor="Black" Margin="0, 0, 20, 0"/>  <Label Text="Обича да спи, пие бира и яде." HorizontalTextAlignment="End" VerticalOptions="Center"  FontSize="Subtitle" FontAttributes="Bold,Italic" TextColor="Black" Margin="0, 0, 20, 0"/>  <Label Text="\*Не съм се снимал в банята, снимката излъчва топлина и става конденз." HorizontalTextAlignment="End" VerticalOptions="Center"  FontSize="Caption" FontAttributes="Bold,Italic" TextColor="Black" Margin="0, 0, 20, 0"/>  </StackLayout>  </Grid>  <Label Grid.Row="1" Text="Цитати:" FontSize="Title" FontAttributes="Bold"  TextColor="Black" Margin="10, 0, 0, 0"/>  <Frame Grid.Row="2" BackgroundColor="#d8ffd8" Margin="10" CornerRadius="15" HeightRequest="160">  <ScrollView VerticalOptions="FillAndExpand" >  <Label Text="{Binding QuotesText}" FontSize="Medium" FontAttributes="Bold"  TextColor="Black" Margin="10, 0, 0, 0" VerticalOptions="FillAndExpand"/>  </ScrollView>  </Frame>  <Label Grid.Row="3" Text="И още много други..." FontSize="Medium" FontAttributes="Italic"  Margin="0, -10, 15, 0" HorizontalTextAlignment="End" HorizontalOptions="End"  HeightRequest="40" MinimumHeightRequest="40"/>  </Grid>  </StackLayout>  </ContentPage> |

Съпровождащият c# файл не е особено интересен. В конструктора се задава себе си като **BindingContext**. Също пази текста на цитатите.

AuthorPage.xaml.cs

|  |
| --- |
| using Xamarin.Forms;  using Xamarin.Forms.Xaml;  namespace App\_9\_2  {  [XamlCompilation(XamlCompilationOptions.Compile)]  public partial class AuthorInfoPage : ContentPage  {  public AuthorInfoPage()  {  InitializeComponent();  BindingContext = this;  }  public string QuotesText => "'- 'Който нож вади, той нож има.'\r\n- 'Имало едно време и сега пак го има.'\r\n- 'Голям залък лапни, ама да не е много голям.'\r\n- 'Където е текло, е мокро.'\r\n- 'Капка по капка прави Капка на квадрат.'";  }  } |

Литература

1. <https://learn.microsoft.com/en-us/xamarin/xamarin-forms/app-fundamentals/shell/flyout>
2. <https://atlas.bspb.org/?s=%D0%A1%D0%B8%D0%BD%D0%B8%D0%B3%D0%B5%D1%80&et_pb_searchform_submit=et_search_proccess&et_pb_include_posts=yes&et_pb_include_pages=yes>
3. <https://atlas.bspb.org/?s=%D0%BB%D0%B5%D1%88%D0%BE%D1%8F%D0%B4&et_pb_searchform_submit=et_search_proccess&et_pb_include_posts=yes&et_pb_include_pages=yes>
4. <https://chat.openai.com/>
5. <https://www.wikipedia.org/>