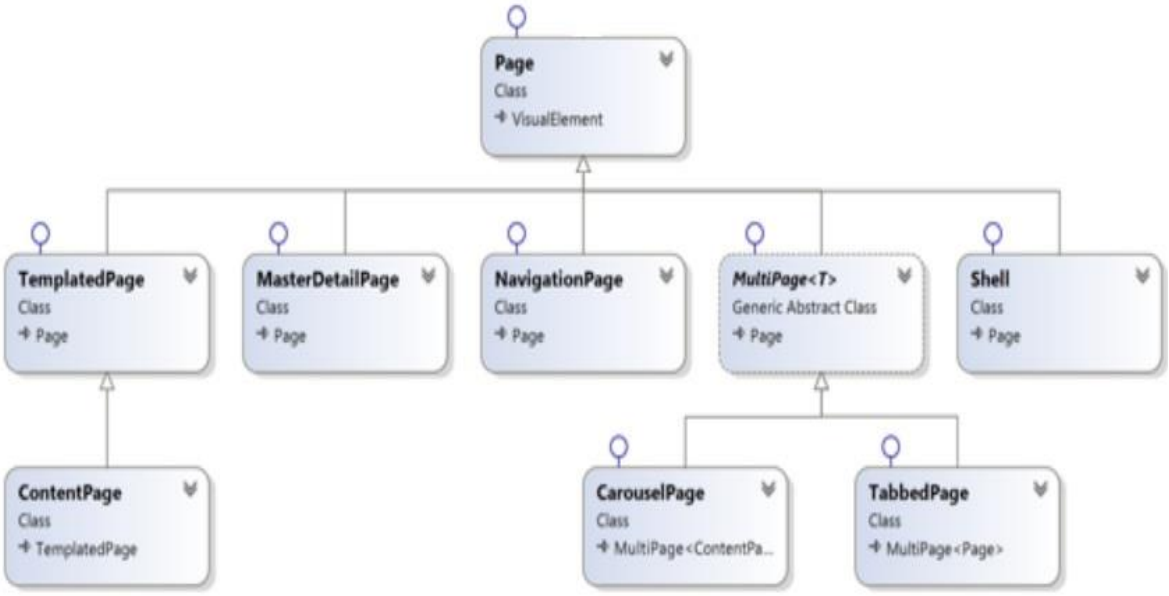
Пользовательский интерфейс: страницы типа ContentPage, FlyoutPage, TabbedPage, CarouselPage и TemplatedPage

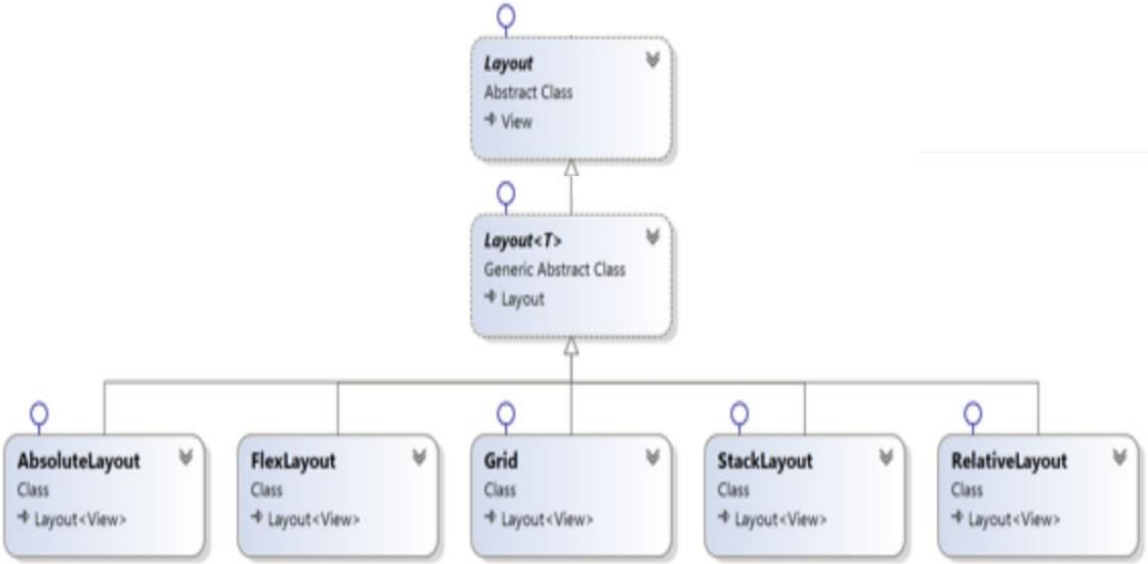
Ниже приведены четыре основные группы элементов, используемых для создания пользовательского интерфейса приложения Xamarin.Forms.

- Страницы
- Макеты
- Элементы управления
- Ячейки

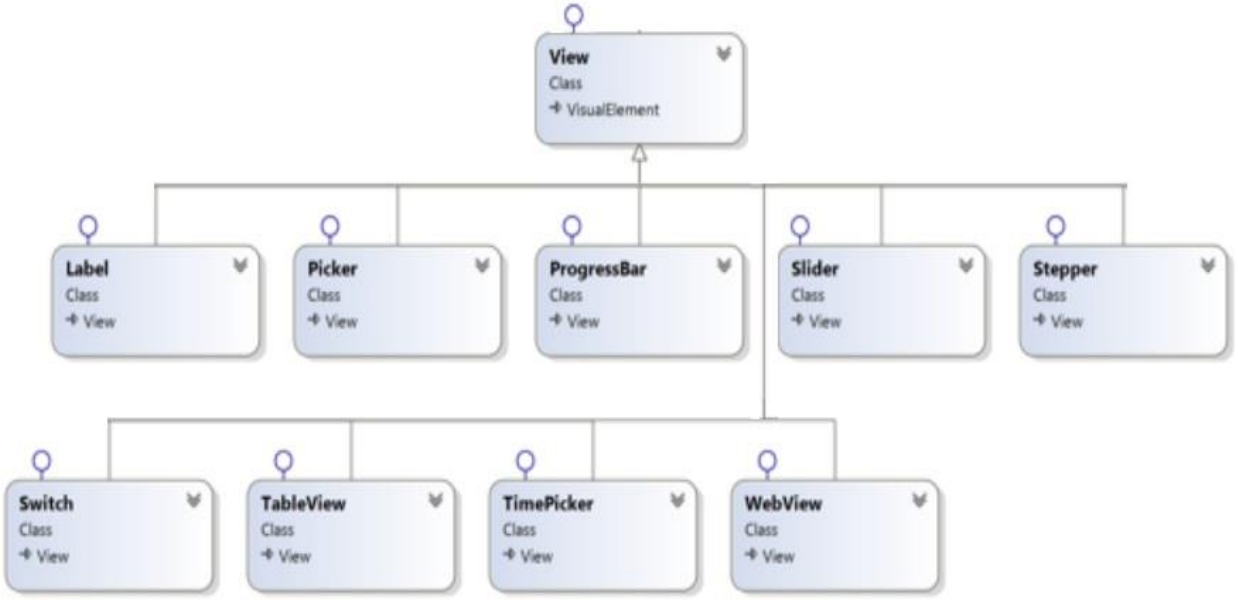
Страница приложения Xamarin.Forms обычно занимает весь экран. Все типы страниц являются производными от класса Page.



Страница обычно содержит макет, который содержит элементы управления и, возможно, другие макеты. Все типы макетов являются производными от класса Layout.

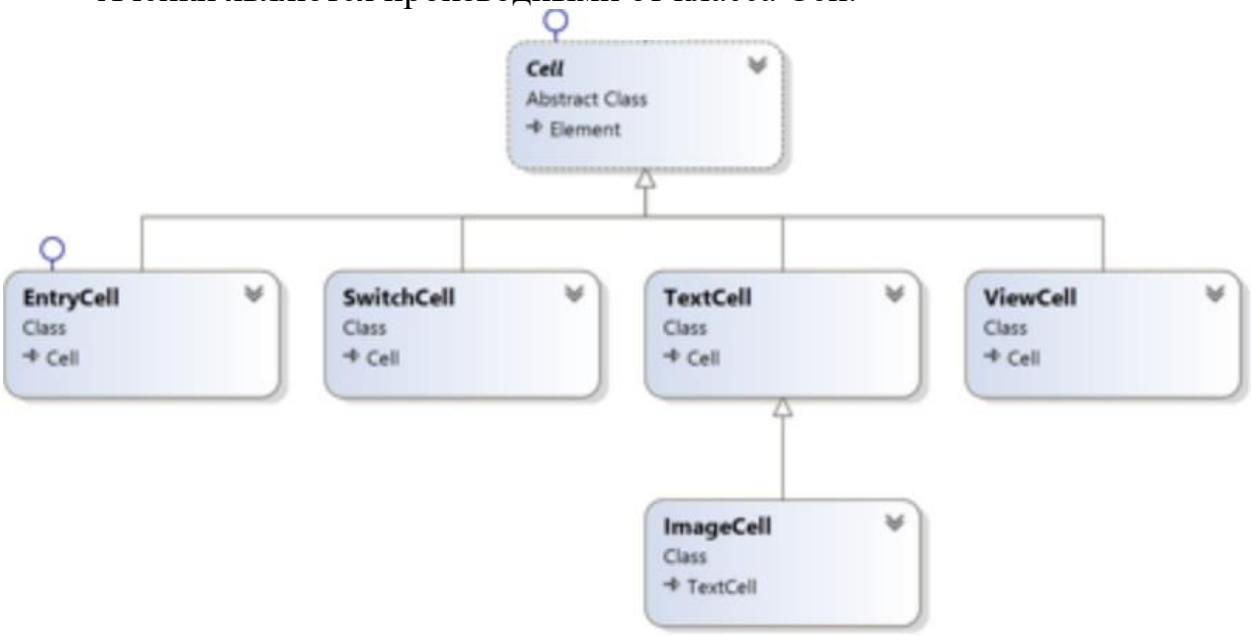


Макет содержит элементы управления или другие макеты. Все типы элементов управления являются производными от класса View.



Ячейки представляют собой специализированные элементы управления, которые используются при отображении данных в TableView и ListView.

Ячейки являются производными от класса Cell.



Страницы, макеты, представления и ячейки в конечном итоге являются производными от класса Element.

Страницы ContentPage, FlyoutPage, NavigationPage, TabbedPage, TempatedPage, CarouselPage представляют собой кроссплатформенные экраны мобильных приложений. Все типы страниц являются производными от класса Page.

Эти визуальные элементы занимают весь экран или большую часть экрана.

В iOS объект класса Page представлен как ViewController, а для Android каждая страница представляет собой Activity, но при этом не является объектом Activity.

Рассмотрим работу со страницами. Страница типа ContentPage - самый простой и наиболее распространенный тип страницы, в которой может содержаться один элемент управления либо один макет.

Для работы со страницей типа ContentPage необходимо создать проект с именем ContentPageDemoPage для мультиплатформенного приложения Xamarin.Forms с использованием шаблона «Пустой».

Удалить автоматически созданный и находящийся в файле MainPage.xaml программный код пользовательского интерфейса.

Для этого в обозревателе решений необходимо щелкнуть правой кнопкой мыши файл MainPage.xaml и выбрать Удалить.

Это действие также удаляет файл программного кода MainPage.xaml.cs.

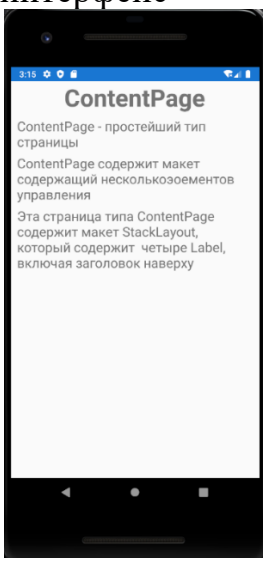
Создать класс в новом файле с именем MainPage.cs. Сделать класс производным от класса ContentPage. Добавьте инструкцию using Xamarin.Forms, так как ContentPage находится в пространстве имен Xamarin.Forms.

```
using System;
using System.Collections.Generic;
using System.Text;
using Xamarin.Forms;

namespace ContentPageDemoPage
{
    Ссылок: 2
    public class MainPage: ContentPage
    {
        ссылка: 1
        public MainPage()
        {
            {
                Label header = new Label
                {
                    Text = "ContentPage",
                    FontSize = 40,
                    FontAttributes = FontAttributes.Bold,
                    HorizontalOptions = LayoutOptions.Center
                };

                Label label1 = new Label
                {
                    Text = "ContentPage - простейший тип страницы",
                    FontSize = Device.GetNamedSize(NamedSize.Large, typeof(Label)),
                };
```

```
            Label label2 = new Label
            {
                Text = "ContentPage содержит макет " +
                       "содержащий несколько" +
                       "эоементов управления",
                FontSize = Device.GetNamedSize(NamedSize.Large, typeof(Label)),
            };

            Label label3 = new Label
            {
                Text = "Эта страница типа ContentPage содержит " +
                   "макет StackLayout, который содержит  " +
                   "четыре Label, включая заголовок наверху",
                FontSize = Device.GetNamedSize(NamedSize.Large, typeof(Label)),
            };

            // Build the page.
            Title = "ContentPage Demo";
            Padding = new Thickness(10, 0);
            Content = new StackLayout
            {
                Children =
                {
                header,
                label1,
                label2,
                label3
                }
                };
            }
        }
    }
}
```

Далее необходимо собрать программное приложение.

При запуске программного приложения должен отобразиться следующий пользовательский интерфейс



Страница FlyoutPage управляет двумя страницами, которые определены свойствами Flyout и Detail. Свойство Flyout используется для задания основной страницы, в которой отображается список или меню. Свойство Detail используется для задания вспомогательной страницы, которая более подробно отображает информацию, которая выбрана на основной странице в списке или меню.

Свойство IsPresented определяет, какая страница отображается – основная (свойство IsPresented равно true) или вспомогательная (свойство IsPresented равно false).

Перейдем к работе со страницей FlyoutPage. для этого создадим проект с именем FPageCS для мультиплатформенного приложения Xamarin.Forms с использованием шаблона «Пустой».

Создадим вспомогательный класс NamedColor, необходимый в дальнейшем для работы с цветами и их названиями, которая выполняется в программном файле MainPage.cs, который будет создан позже. Откройте файл NamedColor.cs и наберите в файле программный код на языке Си Шарп, приведенный на презентации. В приведенном программном коде конструктор класса NamedColor имеет два входных параметра типа string (для приема выдачи названия цвета) и Color (для приема и выдачи непосредственно цвета).

```
using System;
using System.Collections.Generic;
using System.Text;
using Xamarin.Forms;

namespace FPageCS
{
    public class NamedColor
  {
     public NamedColor()
     {
        ;
     }
     public NamedColor(string name, Color color)
     {
        Name = name;
        Color = color;
     }
     public string Name { set; get; }
     public Color Color { set; get; }
     public override string ToString()
     {
        return Name;
     }
  }
}
```

Создадим вспомогательный класс NamedColorPage, необходимый в дальнейшем для работы с вспомогательной страницей, которая соответствует свойству Detail страницы MainPage.

Откройте файл NamedColorPage.cs и наберите в файле программный код на языке Си Шарп, приведенный далее.

В приведенном программном коде формируется страница типа ContentPage, содержащая два макета типа Layout и элемент управления BoxView.

В программном коде сначала формируется элемент управления BoxView, в который с помощью метода SetBinding осуществляется привязка к цвету, который выбран на основной странице в элементе управления ListView.

Далее сформирована функция CreateLabel, предназначенная для формирования элементов шести управления Label, в которых отображаются данные о цвете, выбранном на основной странице в элементе управления ListView.

Далее с помощью свойства Content производится формирование страницы, которая содержит:

макет типа StackLayout, внутри которого находятся три элемента управления Label, в которых отображаются компоненты выбранного на главной странице цвета (red, green, blue), элемент управления;

элемент управления BoxView, в котором отображается выбранный на главной странице цвет;

макет типа StackLayout, содержащий три элемента управления Label, в которых отображаются оттенок или тон цвета (hue), насыщенность, то есть показатель интенсивности тона (saturation), а также яркость по шкале от белого до черного цвета (Luminosity).

```
using System;
using System.Collections.Generic;
using System.Text;
using Xamarin.Forms;

namespace FPageCS
{
    class NamedColorPage: ContentPage
  {
    public NamedColorPage()
    {
      // Элемент управления BoxView для отображения цвета
      BoxView boxView = new BoxView
      {
        WidthRequest = 100,
        HeightRequest = 100,
        HorizontalOptions = LayoutOptions.Center
      };
      boxView.SetBinding(BoxView.ColorProperty, "Color");
```

```
        // Функция для формирования шести элементов управления Label
        Func<string, string, Label> CreateLabel = (string source, string fmt) =>
        {
          Label label = new Label
          {
            FontSize = Device.GetNamedSize(NamedSize.Large, typeof(Label)),
            HorizontalTextAlignment = TextAlignment.End
          };
          label.SetBinding(Label.TextProperty,
            new Binding(source, BindingMode.OneWay, null, null, fmt));

          return label;
        };

        // Построение страницы (макет Stacklayout, BoxView, макет Stacklayout)
        Content = new StackLayout
        {
          Children =
          {
            new StackLayout
            {
              HorizontalOptions = LayoutOptions.Center,
              VerticalOptions = LayoutOptions.CenterAndExpand,
              Children =
              {
                CreateLabel ("Color.R", "R = {0:F2}"),
                CreateLabel ("Color.G", "G = {0:F2}"),
                CreateLabel ("Color.B", "B = {0:F2}"),
              }
            },
            boxView,
            new StackLayout
            {
              HorizontalOptions = LayoutOptions.Center,
              VerticalOptions = LayoutOptions.CenterAndExpand,
              Children =
              {
                CreateLabel ("Color.Hue", "Hue = {0:F2}"),
                CreateLabel ("Color.Saturation", "Saturation = {0:F2}"),
                CreateLabel ("Color.Luminosity", "Luminosity = {0:F2}")
              }
            }
          }
        };
      }
    }
}
```

Будем формировать пользовательский интерфейс с помощью программного кода Си Шарп.

Для этого сначала необходимо удалить автоматически созданный и находящийся в файле MainPage.xaml программный код пользовательского интерфейса. Для этого в обозревателе решений необходимо щелкнуть правой кнопкой мыши файл MainPage.xaml и выбрать Удалить. Это действие также удаляет и файл программного кода MainPage.xaml.cs.

Далее необходимо создать класс MainPage, для чего необходимо подвести курсор к проекту без суффикса, нажать правую кнопку мыши, выбрать меню Добавить, выбрать Класс и в нижней части диалогового окна набрать имя класса MainPage.cs. В результате будет создан файл с именем MainPage.cs.

Откройте файл MainPage.cs.

Добавьте инструкцию using Xamarin.Forms в верхней части программного кода, так как FlyoutPage находится в пространстве имен Xamarin.Forms.

В программном коде сделайте класс MainPage производным от класса FlyoutPage. Для этого после имени класса MainPage поставьте двоеточие и введите FlyoutPage.

Сначала создается элемент управления Label с именем header, в котором размещается заголовок страницы - текст «FlyoutPage Demo».

Затем формируется массив namedColors, содержащий цвета и их названия.

Названия цветов будут отображаться в виде списка в элементе управления ListView, который будет располагаться на основной странице, то есть, странице, которая соответствует свойству Flyout.

Далее создается элемент управления ListView.

Объект ListView заполняется данными с помощью свойства ItemsSource, которое определяет, что в качестве элементов списка в элементе управления ListView будут использованы названия цветов из массива namedColors.

Свойство Margin элемента управления ListView задает расстояние между краями страницы и краями ListView.

Расстояние задается с использованием структуры Thickness, которая в программном коде имеет два параметра, что соответствует заданию расстояния по горизонтали и вертикали.

Горизонтальное значение будет симметрично применено к левой и правой сторонам ListView, а вертикальное значение будет симметрично применено к верхней и нижней сторонам элемента управления ListView

Далее с помощью свойства Flyout формируется основная страница типа ContentPage названием «Color List».

С помощью свойства Content в страницу FlyoutPage помещается макет типа StackLayout, в котором с помощью свойства Children размещаются сформированные ранее элементы управления Label с именем header, а также элемент управления ListView.

Далее с помощью свойства Detail страницы MainPage создается вспомогательная страница detailPage как экземпляр класса NamedColorPage, программный код которого рассматривался ранее.

Вспомогательная страница отображает более подробно информацию о цвете, выбранном в списке цветов, отображаемом в элементе управления ListView в главной странице.

Далее приведен программный код, который производит обработку события ItemSelected при выборе цвета в списке, отображаемом в элементе управления ListView.

При выборе цвета происходит извлечение объекта SelectedItem в качестве аргумента события ItemSelected.

Параметр sender события ItemSelected соответствует элементу управления ListView, внутри которого производился выбор цвета.

Вспомогательной странице detailPage, являющейся экземпляром класса NamedColorPage, с помощью свойства Detail передается информация о выбранном цвете (BindingContext).

За отображение основной или вспомогательной страницы отвечает свойство IsPresented страницы MainPage типа FlyoutPage.

Если при запуске программного приложения на экране мобильного устройства необходимо первым отобразить основную страницу, то свойству IsPresented присваивается значение true, а если вспомогательную, то свойству IsPresented присваивается значение false.

В зависимости от вида мобильного устройства, на котором выполняется программное приложение и ориентации экрана мобильного устройства применяется несколько способов управления перехода от главной страницы к вспомогательной.

За управление переходом между страницами отвечает свойство **FlyoutLayoutBehavior**, которое может принимать несколько значений:

- Default — страницы отображаются по умолчанию в соответствии с используемой платформой
- Popover — вспомогательная страница частично перекрывает главную страницу.
- Split — главная страница отображается слева, а вспомогательная страница отображается справа.
- SplitOnLandscape — экран мобильного устройства разделяется на две части в случае, если ориентация экрана альбомная
- SplitOnPortrait — экран мобильного устройства разделяется на две части в случае, если ориентация экрана книжная

```
using System;
using System.Collections.Generic;
using System.Text;
using Xamarin.Forms;

namespace FPageCS
```

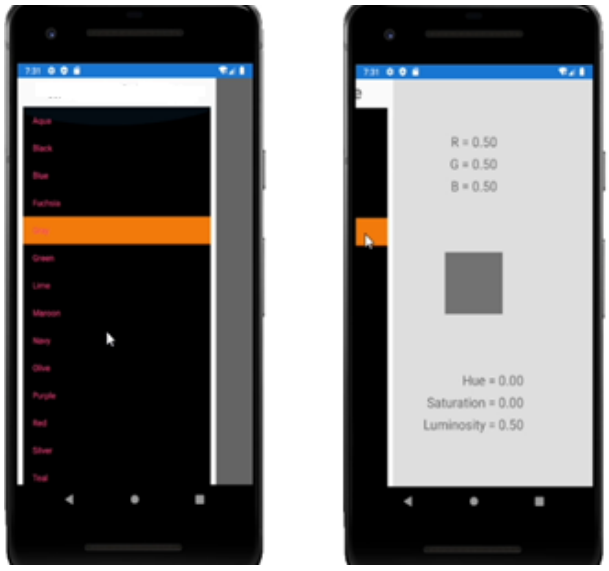
```
{
   class MainPage: FlyoutPage
   {
      public MainPage()
      {
         Title = "FlyoutPage Demo";

         Label header = new Label
         {
            Text = "FlyoutPage",
            FontSize = 30,
            FontAttributes = FontAttributes.Bold,
            HorizontalOptions = LayoutOptions.Center
         };

         // Массив объектов NamedColor
         NamedColor[] namedColors = {
            new NamedColor ("Aqua", Color.Aqua),
            new NamedColor ("Black", Color.Black),
            new NamedColor ("Blue", Color.Blue),
            new NamedColor ("Fuchsia", Color.Fuchsia),
            new NamedColor ("Gray", Color.Gray),
            new NamedColor ("Green", Color.Green),
            new NamedColor ("Lime", Color.Lime),
            new NamedColor ("Maroon", Color.Maroon),
            new NamedColor ("Navy", Color.Navy),
            new NamedColor ("Olive", Color.Olive),
            new NamedColor ("Purple", Color.Purple),
            new NamedColor ("Red", Color.Red),
            new NamedColor ("Silver", Color.Silver),
            new NamedColor ("Teal", Color.Teal),
            new NamedColor ("White", Color.White),
            new NamedColor ("Yellow", Color.Yellow)
         };

         // Создание элемента управления ListView для flyout page.
         ListView listView = new ListView
         {
            ItemsSource = namedColors,
            Margin = new Thickness(10, 0),
            BackgroundColor=Color.Black,

         };

         // Формирование flyout page с помощью элемента управления ListView.
         Flyout = new ContentPage
         {
            Title = "Color List",
            Content = new StackLayout
```

```
            {
                Children = {
                    header,
                    listView
                }
            }
        };

        // Создание detail page с использованием класса NamedColorPage
        NamedColorPage detailPage = new NamedColorPage();
        Detail = detailPage;

        // Define a selected handler for the ListView.
        listView.ItemSelected += (sender, args) => {
            // Set the BindingContext of the detail page.
            Detail.BindingContext = args.SelectedItem;
            // Show the detail page.
            IsPresented = true;
        };
        // Initialize the ListView selection.
        listView.SelectedItem = namedColors[5];
        FlyoutLayoutBehavior = FlyoutLayoutBehavior.Popover;
    }
  }
}
```

После запуска программного приложения отображается основная страница, на которой отображен список цветов, который снабжен возможностью перемотки. Сначала в списке цветов выбран серый цвет.

Чтобы посмотреть вспомогательную страницу, на которой приведены более подробные сведения о выбранном цвете, необходимо сдвинуть влево основную страницу.

Далее выбираем красный цвет и также для просмотра вспомогательной страницы выполняем сдвиг влево основной страницы.



Далее рассмотрим работу со страницей типа NavigationPage, с помощью которой производится управление навигацией между страницами (диалоговыми окнами) с помощью архитектуры на основе стека. При навигации по страницам в программном приложении экземпляр страницы должен передаваться конструктору объекта NavigationPage. Класс NavigationPage обеспечивает иерархическую навигацию, при которой пользователь может переходить по страницам вперед и назад. Класс NavigationPage реализует навигацию на основе стека объектов Page по методу LIFO (последним поступил— первым обслужен). При переходе с одной страницы на другую программное приложение помещает новую страницу (на которую происходит переход) в стек навигации, где она помещается в вершину и становится активной страницей.

Для возврата от текущей к предыдущей странице приложение убирает текущую страницу из вершины стека навигации, после чего активной становится та страница, которая оказалась на вершине стека.

Методы навигации предоставляются свойством Navigation для страниц любых типов, производных от класса Page.

При иерархической навигации класс NavigationPage используется для перехода по стеку, состоящего из объектов ContentPage. Первая страница, добавленная в стек навигации, называется корневой страницей программного приложения.

Для работы со страницей типа NavigationPage создается проект с именем NavPage для мультиплатформенного приложения Xamarin.Forms с использованием шаблона «Пустой». Формирование пользовательского интерфейса будет производится с помощью программного кода Си Шарп. Для этого сначала необходимо удалить автоматически созданный и находящийся в файле MainPage.xaml программный код пользовательского интерфейса. Для этого в обозревателе решений необходимо щелкнуть правой кнопкой мыши по файлу MainPage.xaml и выбрать меню «Удалить». После этого также удаляется и файл MainPage.xaml.cs, связанный с файлом MainPage.xaml.

Далее необходимо открыть в проекте файл App.xaml.cs и добавить в него программный код, который добавляет корневую страницу Page1 в стек навигации с помощью класса NavigationPage. В результате страница Page1

помещается в вершину стека, становится активной и отображается первой при запуске приложения.

```
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace NavPage
{
   public partial class App: Application
   {
      public App()
      {
        MainPage = new NavigationPage(new Page1());
      }
   }
}
```

Далее создается страница LabelPage, к которой будет производиться переход от страницы Page1. Также со страницы Page1 будет производиться переход к странице BoxViewPage.

Создается класс LabelPage с использованием действий по созданию класса, рассмотренных ранее. Далее необходимо открыть созданный файл LabelPage.cs и сделайть класс LabelPage дочерним от класса ContentPage. Так как страницы ContentPage и NavigationPage поддерживаются Xamarin.Forms, то необходимо добавить инструкцию using Xamarin.Forms в верхней части программного кода для доступа к соответствующему пространству имен.

Далее необходимо набрать в файле программный код на языке Си Шарп, приведенный ниже.

Страница содержит макет типа StackLayout с двумя метками и двумя кнопками для перехода к странице BoxViewPage и возврата на страницу Page1.

Переход к странице BoxViewPage производится при нажатии на кнопку NextPage. Обработчик события, происходящего при нажатии на кнопку NextPage, вызывает метод PushAsync свойства Navigation страницы LabelPage. Возврат к странице Page1 производится при нажатии на кнопку PrevPage. Обработчик события, происходящего при нажатии на кнопку PrevPage, вызывает метод PopAsync.

В результате страница LabelPage удаляется из стека навигации, а активной становится верхняя страница в стеке, то есть, Page1.

```
namespace NavPage
{
   class LabelPage: ContentPage
   {
      public LabelPage()
      {
         Label header = new Label
         {
```

```
    Text = "Label",
    FontSize = 50,
    FontAttributes = FontAttributes.Bold,
    HorizontalOptions = LayoutOptions.Center
};

Label label = new Label
{
   Text = "Xamarin.Forms позволяет создавать " +
 "пользовательские интерфейсы, являющиеся " +
 "общими для платворм Android и iOS",
   FontSize = Device.GetNamedSize(NamedSize.Large,
                                  typeof(Label)),
    VerticalOptions = LayoutOptions.CenterAndExpand,
    Margin = new Thickness(10, 0)
};

Button nextPageButton = new Button
{ Text = "Next Page",
  VerticalOptions = LayoutOptions.CenterAndExpand
};
nextPageButton.Clicked += NextPage;

Button previousPageButton = new Button
{ Text = "Previous Page",
  VerticalOptions = LayoutOptions.CenterAndExpand
};
previousPageButton.Clicked += PrevPage;

// Построение страницы
Title = "Label Demo";
Content = new StackLayout
{
    Children =
    {
        header,
        label,
        nextPageButton,
        previousPageButton
    }
};

async void NextPage(object sender, EventArgs e)
{
    await Navigation.PushAsync(new BoxViewPage());
```

```
        }
        async void PrevPage(object sender, EventArgs e)
        {
            await Navigation.PopAsync();
        }
    }
  }
}
```

Создается страница BoxViewPage, к которой будет производиться переход от страниц Page1 и LabelPage. Для этого необходимо создать класс BoxViewPage, открыть созданный файл BoxViewPage.cs и сделать класс BoxViewPage дочерним от класса ContentPage.

Далее необходимо добавить инструкцию using Xamarin.Forms в верхней части программного кода. После этого необходимо набрать в файле программный код на языке Си Шарп, приведенный ниже.

Страница содержит макет типа StackLayout с элементами управления Label и BoxView и кнопкой для возврата к страницам LabelPage и Page1.

Возврат к предыдущим страницам производится при нажатии на кнопку NextPage. При этом вызывается метод PopAsync. В результате страница BoxViewPage удаляется из стека навигации, а активной становится верхняя страница в стеке, то есть, Page1 или LabelPage.

```
namespace NavPage
{
  class BoxViewPage: ContentPage
  {
    public BoxViewPage()
    {
      Label header = new Label
      {
        Text = "BoxView",
        FontSize = 50,
        FontAttributes = FontAttributes.Bold,
        HorizontalOptions = LayoutOptions.Center
      };
      BoxView boxView = new BoxView
      {
        Color = Color.Orange,
        WidthRequest = 200,
        HeightRequest = 200,
        HorizontalOptions = LayoutOptions.Center,
        VerticalOptions = LayoutOptions.CenterAndExpand
      };
      Button previousPageButton = new Button
      {
        Text = "Previous Page",
```

```
            VerticalOptions = LayoutOptions.CenterAndExpand
        };
        previousPageButton.Clicked += PrevPage;
        // Построение страницы
        Title = "BoxView Demo";
        Content = new StackLayout
        {
            Children =
            {
                header,
                boxView,
                previousPageButton
            }
        };
        async void PrevPage(object sender, EventArgs e)
        {
            await Navigation.PopAsync();
        }
    }
  }
}
```

Перейдем к созданию страницы Page1. Создайте класс Page1, откройте созданный файл Page1.cs и сделайте класс Page1 дочерним от класса ContentPage.

Добавьте инструкцию using Xamarin.Forms в верхней части программного кода, так как ContentPage находится в пространстве имен Xamarin.Forms.

Наберите в файле программный код на языке Си Шарп, приведенный на презентации. Внутри страницы расположен макет с двумя кнопками для перехода к страницам LabelPage и BoxViewPage.

Также в программном коде сформированы обработчики событий нажатия на кнопки. Обработчики используют метод PushAsync свойства Navigation страницы Page1 для перехода к страницам LabelPage и BoxViewPage.

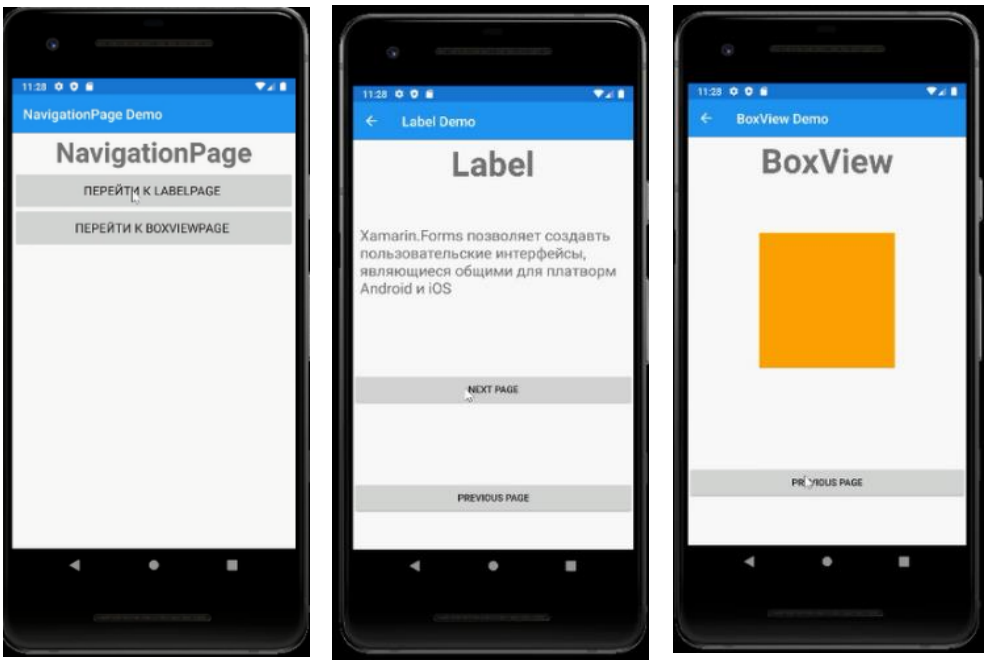
```
namespace NavPage
{
  class Page1: ContentPage
  {
    public Page1()
    {
      Label header = new Label
      {
        Text = "NavigationPage",
        FontSize = 40,
        FontAttributes = FontAttributes.Bold,
        HorizontalOptions = LayoutOptions.Center
      };
```

```
            Button button1 = new Button
            {
               Text = "Перейти к LabelPage ",
               Font = Font.SystemFontOfSize(NamedSize.Large),
               BorderWidth = 1,
               WidthRequest=400
            };
            button1.Clicked += async (sender, args) =>
               await Navigation.PushAsync(new LabelPage());

            Button button2 = new Button
            {
               Text = "Перейти к BoxViewPage ",
               Font = Font.SystemFontOfSize(NamedSize.Large),
               BorderWidth = 1,
               WidthRequest = 400
            };
            button2.Clicked += async (sender, args) =>
               await Navigation.PushAsync(new BoxViewPage());
            // Построение страницы
            Title = "NavigationPage Demo";
            Content = new StackLayout
            {
               Children =
               {
                  header,
                  new StackLayout
                  {
                     HorizontalOptions = LayoutOptions.Center,
                     Children =
                     {
                        button1,
                        button2
                     }
                  }
               }
            };
         }
      }
   }
```

При запуске программного приложения на экране мобильного устройства отображается страница Page1 с двумя кнопками «Перейти к LabelPage» и «Перейти к BoxViewPage».

При нажатии на кнопку «Перейти к LabelPage» происходит переход к странице LabelPage, с которой можно перейти к странице BoxViewPage или

вернуться обратно к корневой странице Page1. В случае перехода на страницу BoxViewPage можно произвести возврат к странице LabelPage.



Далее рассмотрим работу со страницей **TabbedPage**, которая является дочерней от абстрактного класса MultiPage и позволяет перемещаться между страницами с помощью списка вкладок.

В iOS список вкладок отображается в нижней части экрана, а также в области данных сверху.

У каждой вкладки есть заголовок, а также может быть значок, который должен быть PNG-файлом.

В книжной ориентации значки панели вкладок отображаются над заголовками вкладок.

В альбомной ориентации значки и заголовки отображаются рядом.

Кроме того, в зависимости от устройства и ориентации может отображаться обычная или компактная панель вкладок.

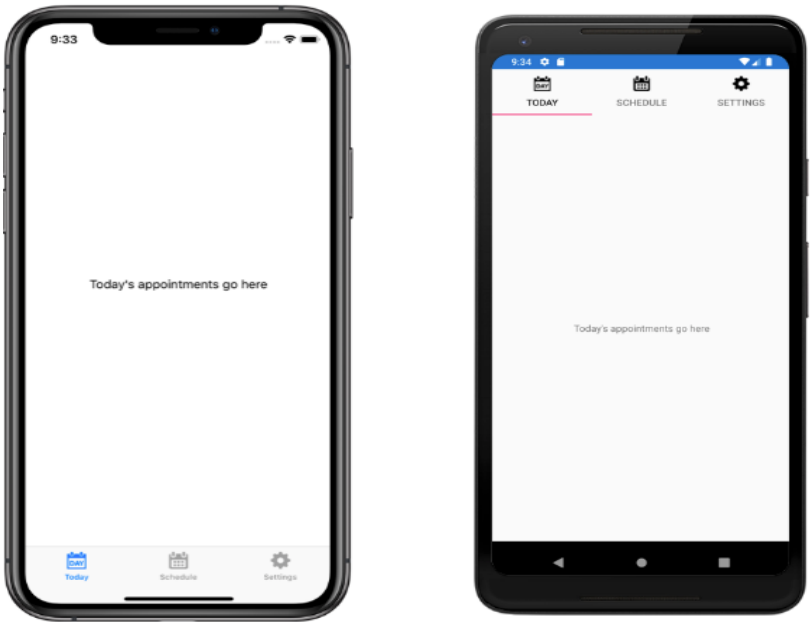
При наличии более пяти вкладок появляется вкладка «Дополнительно», используемая для доступа к дополнительным вкладкам.

В Android список вкладок отображается в верхней части экрана, а также в области данных ниже.

У каждой вкладки есть заголовок и значок, который должен быть PNG-файлом с альфа-каналом.

Но эти вкладки можно переместить в нижнюю часть экрана в зависимости от конкретной платформы.

При наличии более пяти вкладок, список которых отображается внизу экрана, появляется вкладка «Дополнительно», которую можно использовать для доступа к дополнительным вкладкам.



Создать страницу TabbedPage можно двумя способами:

1. Производится заполнение TabbedPage коллекцией дочерних объектов типа Page, например, страницами типа ContentPage.

2. Производится назначение коллекции страниц Page свойству ItemsSource страницы TabbedPage, а затем назначить шаблон DataTemplate свойству ItemTemplate для однообразного отображения страниц, входящих в состав коллекции.

При обоих подходах TabbedPage отображает страницу, которая соответствует выбранной вкладке.

Страница TabbedPage имеет следующие свойства:

BarBackgroundColor (тип Color) — цвет фона панели вкладок;

BarTextColor (тип Color) — цвет текста на панели вкладок;

SelectedTabColor (тип Color) — цвет выбранной вкладки;

UnselectedTabColor (тип Color) — цвет вкладки, если она не выбрана.

Создадим проект TabPage для мультиплатформенного проекта Xamarin.Forms с использованием шаблона «Пустой». Откроем файл MainPage.xaml.cs и скорректируем программный код в соответствии с презентацией. Из данного файла видно, что страница MainPage является страницей типа TabbedPage.

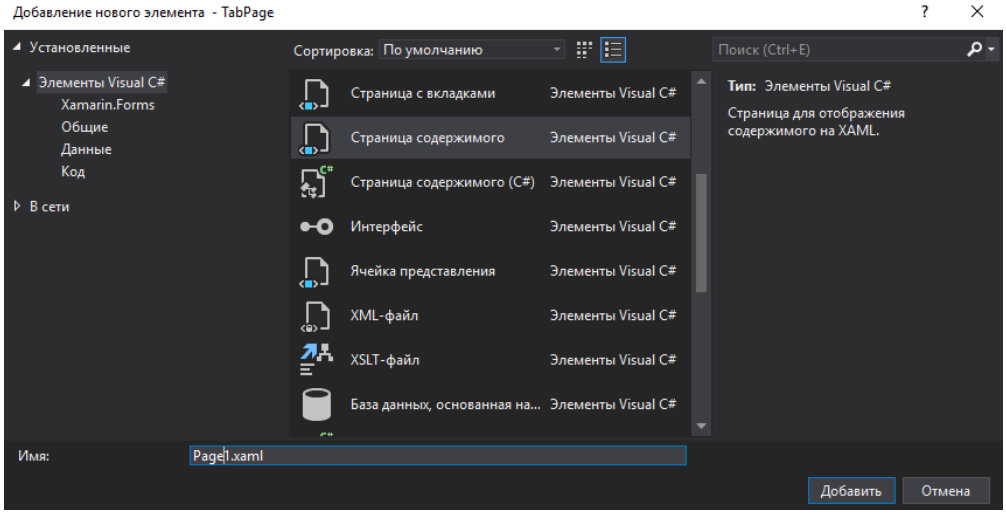
```
using Xamarin.Forms;
namespace TabPage
{
	public partial class MainPage : TabbedPage
	{
		public MainPage ()
		{
			InitializeComponent ();
		}
	}
}
```

Откроем файл MainPage.xaml и наберем программный код, приведенный в презентации.

В программном коде XAML страницы MainPage в коллекцию TabbedPage с помощью свойства Children добавлены страницы Page1, Page2 и Page3.

```
<TabbedPage xmlns="http://xamarin.com/schemas/2014/forms"
        xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        xmlns:local="clr-namespace:TabPage;assembly=TabPage"
        x:Class="TabPage.MainPage">
  <TabbedPage.Children>
    <local:Page1 />
    <local:Page2 />
    <local:Page3 />
  </TabbedPage.Children>
</TabbedPage>
```

Добавим страницу Page1 в проект. Для этого необходимо подвести курсор мыши к общему проекту TabPage, щелкните правой кнопкой мыши и выберите «Добавить» - «Новый элемент». В диалоговом окне «Добавление нового элемента» выберите «Страница содержимого», которая создает страницу типа ContentPage. Присвойте странице имя Page1.



Необходимо открыть файл Page1.xaml и набрать программный код xaml страницы Page1, на которую производится ссылка из программного кода страницы MainPage. В странице расположен макет типа StackLayout, внутри которого находятся элемент управления Label с поясняющим текстом и элемент управления BoxView оранжево-красного цвета.

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
        xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        xmlns:d="http://xamarin.com/schemas/2014/forms/design"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        mc:Ignorable="d"
```

```
        x:Class="TabPage.Page1"
        BackgroundColor="Coral"
        Title=" BoxView ">
  <ContentPage.Content>
    <StackLayout>
      <Label Text="BoxView предназначен для
          отображения прямоугольника с заданной
          шириной, высотой и цветом"
        VerticalTextAlignment="Center"
        HorizontalTextAlignment="Center"
        WidthRequest="300"   BackgroundColor="Brown"
        HeightRequest="300"  TextColor="Yellow"
        FontAttributes="Bold"
        FontSize="30"
         />
      <BoxView
        WidthRequest="250"
        HeightRequest="250"
        VerticalOptions="CenterAndExpand"
        HorizontalOptions="Center"
        BackgroundColor="OrangeRed"/>
    </StackLayout>
  </ContentPage.Content>
</ContentPage>
```

Добавим страницу Page2 в проект. Необходимо открыть файл Page2.xaml и набрать программный код xaml страницы Page2, на которую производится ссылка из программного кода страницы MainPage. В странице расположен макет типа StackLayout, внутри которого находятся элемент управления Label с поясняющим текстом и элемент управления Editor цвета «Bisque».

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
        xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        xmlns:d="http://xamarin.com/schemas/2014/forms/design"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        mc:Ignorable="d"
        x:Class="TabPage.Page2"
        BackgroundColor="DarkGray"
        Title="Editor">
  <ContentPage.Content>
    <StackLayout>

      <Label Text="Editor позволяет пользователю
          вводить и редактировать несколько
          строк текста"
        VerticalTextAlignment="Center"
        HorizontalTextAlignment="Center"
        WidthRequest="300"   BackgroundColor="Brown"
```

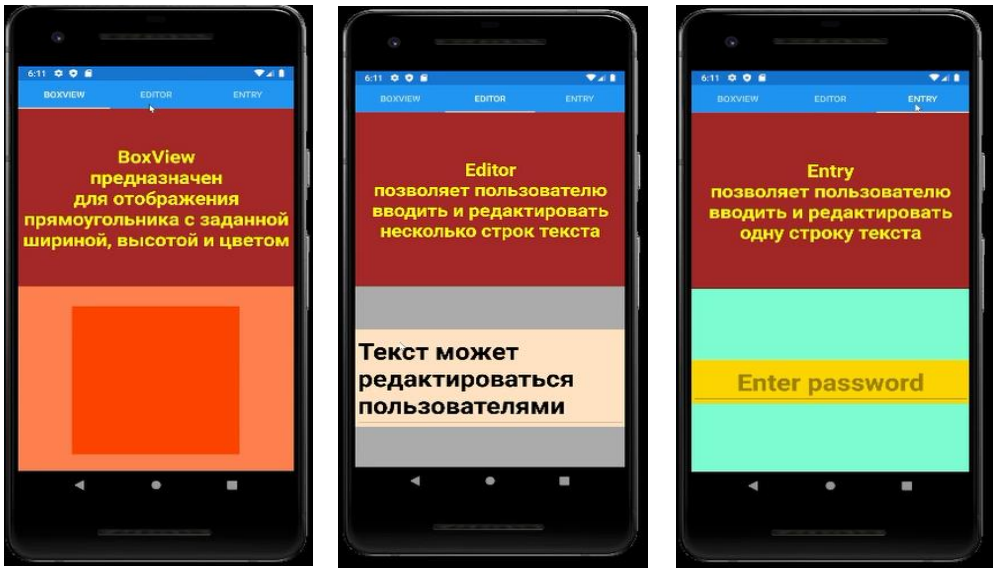
```
        HeightRequest="300"  TextColor="Yellow"
        FontAttributes="Bold"
        FontSize="30"/>

      <Editor
        BackgroundColor="Bisque"
        VerticalOptions="CenterAndExpand"
                TextColor="Black"
        Text="Текст может редактироваться пользователями"
        FontSize="40" FontAttributes="Bold"/>
    </StackLayout>
  </ContentPage.Content>
</ContentPage>
```

Добавим страницу Page3 в проект. Необходимо открыть файл Page3.xaml и набрать программный код xaml страницы Page3, на которую производится ссылка из программного кода страницы MainPage. В странице расположен макет типа StackLayout, внутри которого находятся элемент управления Label с поясняющим текстом и элемент управления Entry цвета «Gold».

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
        xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        xmlns:d="http://xamarin.com/schemas/2014/forms/design"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        mc:Ignorable="d"
        x:Class="TabPage.Page3"
        BackgroundColor="Aquamarine"
        Title="Entry">
  <ContentPage.Content>
    <StackLayout>

      <Label Text="Entry позволяет пользователю
          вводить и редактировать одну строку
          текста."
        VerticalTextAlignment="Center"
        HorizontalTextAlignment="Center"
        WidthRequest="300"   BackgroundColor="Brown"
        HeightRequest="300"  TextColor="Yellow"
        FontAttributes="Bold"
        FontSize="30"/>

      <Entry Keyboard="Text"
        BackgroundColor="Gold"
        HorizontalTextAlignment="Center"
        Placeholder="Enter password"
        VerticalOptions="CenterAndExpand"
        FontSize="40" FontAttributes="Bold"/>
```

```
    </StackLayout>
  </ContentPage.Content>
</ContentPage>
```

Запустим программное приложение. Отображается страница с тремя вкладками. Имеется возможность переключаться между вкладками, в которых отображаются названия элементов управления, используемых для формирования пользовательского интерфейса, и поясняется назначение элементов управления.



Страница **CarouselPage** в Xamarin.Forms позволяет перемещаться между страницами с помощью жеста прокрутки, то есть когда пользователь для перелистывания страниц проводит пальцем по сенсорному экрану вправо для перехода вперед по коллекции страниц или влево для перехода назад.

Если объект CarouselPage внедрен на страницу Detail страницы FlyoutPage, свойству FlyoutPage.IsGestureEnabled необходимо присвоить значение false, чтобы предотвратить конфликты между страницами CarouselPage и FlyoutDetailPage.

Создать CarouselPage можно двумя способами:

1. Необходимо свойству Children страницы CarouselPage назначить коллекцию страниц. При этом в коллекцию могут входить только страницы типа ContentPage или объекты типа ContentPage.

2. Назначить свойству ItemsSource страницы CarouselPage коллекцию страниц. Затем свойству ItemTemplate присвоить DataTemplate (описание шаблона, задающего одинаковое отображение каждой страницы из коллекции, указанной в свойстве ItemsSource).

Оба способа приводят к тому, что страницы, входящие в коллекцию страницы CarouselPage, будут отображаться по очереди при проведении пальцем по экрану.

Создадим проект CarPage для мультиплатформенного проекта Xamarin.Forms с использованием шаблона «Пустой». Откроем файл MainPage.xaml.cs и скорректируем программный код в соответствии с

презентацией. Из данного файла видно, что страница MainPage является страницей типа CarouselPage.

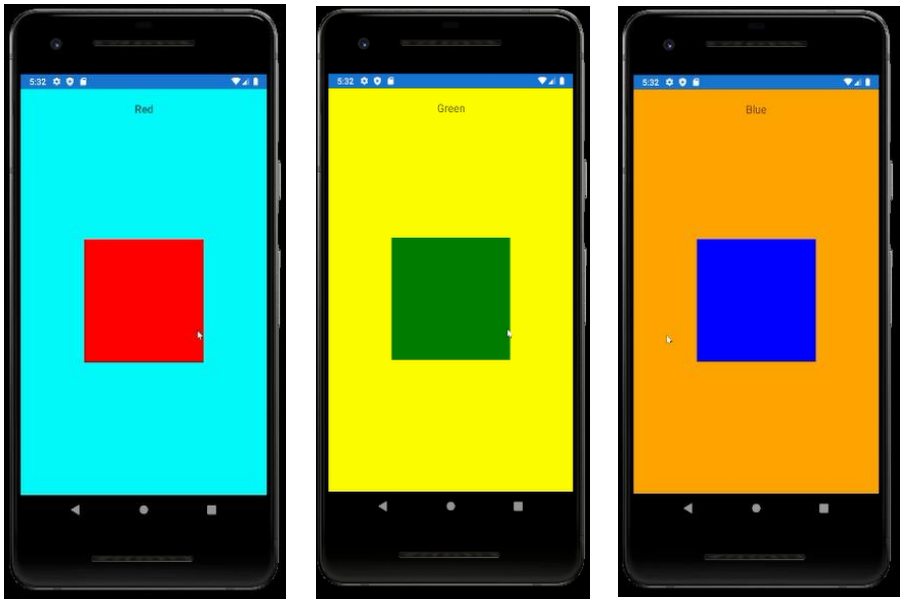
```
using Xamarin.Forms;
namespace CarPage
{
        public partial class MainPage : CarouselPage
        {
                public MainPage ()
                {
                        InitializeComponent ();
                }
        }
}
```

Откроем файл MainPage.xaml. В этом файле приведен программный код XAML, который соответствует странице CarouselPage, содержащей коллекцию из трех страниц типа ContentPage. В данном программном приложении страница CarouselPage создается первым из указанных выше способов (то есть создается коллекция страниц типа ContentPage). В коллекцию входят три страницы типа ContentPage. Каждая страница содержит макет типа StackLayout, внутри которого находятся элементы управления Label и BoxView. На первой странице цвет фона "Cyan" и элемент управления BoxView красного цвета. На второй странице желтый цвет фона, а элемент управления BoxView зеленого цвета. На третьей странице оранжевый цвет фона, а элемент управления BoxView синего цвета.

```
<?xml version="1.0" encoding="UTF-8"?>
<CarouselPage xmlns="http://xamarin.com/schemas/2014/forms"
         xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
                x:Class="CarPage.MainPage">
  <ContentPage
    BackgroundColor="Cyan" Padding="20">
    <StackLayout>
      <Label Text="Red"
          FontSize="Medium" FontAttributes="Bold"
          HorizontalOptions="Center" />
      <BoxView Color="Red"
           WidthRequest="200"    HeightRequest="200"
           HorizontalOptions="Center"
           VerticalOptions="CenterAndExpand" />
    </StackLayout>
  </ContentPage>
  <ContentPage
    BackgroundColor="Yellow" Padding="20">
    <StackLayout>
      <Label Text="Green"
          FontSize="Medium"
          HorizontalOptions="Center" />
      <BoxView Color="Green"
```

```
            WidthRequest="200"   HeightRequest="200"
            HorizontalOptions="Center"
            VerticalOptions="CenterAndExpand" />
    </StackLayout>
  </ContentPage>
  <ContentPage
    BackgroundColor="Orange" Padding="20">
    <StackLayout>
      <Label Text="Blue"
          FontSize="Medium"
          HorizontalOptions="Center" />
      <BoxView Color="Blue"
          WidthRequest="200"    HeightRequest="200"
          HorizontalOptions="Center"
          VerticalOptions="CenterAndExpand" />
    </StackLayout>
  </ContentPage>
</CarouselPage>
```

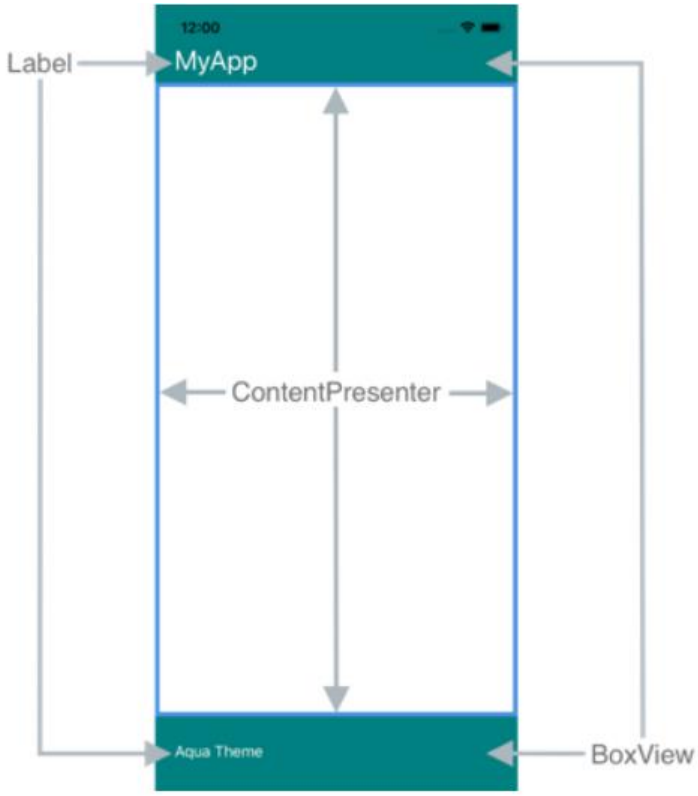
После запуска программного приложения отображается первая страница с элементом управления BoxView с заливкой красного цвета. При перелистывании будут появляться страницы с элементами управления BoxView зеленого и синего цвета



Страница **TemplatedPage** позволяет определить шаблон для задания стандартной визуальной структуры для страниц типа ContentPage. Таким образом, шаблон страницы можно использовать для одинакового отображения нескольких страниц. В этом случае элементы управления в страницах будут отображаться однообразно с учетом настроек, содержащихся в шаблоне.

Для шаблонной страницы, внутри которой содержатся несколько элементов управления, используется шаблон ControlTemplate, который задает

элементы управления, входящие в состав шаблона. В шаблоне можно задать элемент управления ContentPresenter, в котором размещаются элементы управления, передаваемые шаблону со страницы, к которой применяется шаблон. На рисунке показан шаблон ControlTemplate для страницы, содержащий элементы управления Label, элементы управления BoxView, а также элемент ContentPresenter в виде прямоугольника.



Создадим проект TemPage для мультиплатформенного проекта Xamarin.Forms с использованием шаблона «Пустой». Создадим файл HeaderFooterPage.xaml, в котором заданы шаблоны «TealTemplate» для формирования главной страницы MainPage, программный код которой обращается к шаблону и будет создан позже. Наберем программный код в соответствии с презентацией.

В начале страницы устанавливаются настройки для элементов управления Entry, расположенных в странице MainPage. Настройки устанавливаются с помощью Style и свойства TargetType. Производится настройка стиля визуального отображения элементов управления в случае работы с ними (то есть, при переходе к ним фокуса).

Обычно шаблон ControlTemplate объявляют в качестве ресурса. Если шаблон ControlTemplate объявлен как ресурс, он должен иметь ключ, заданный с помощью атрибута x:Key.

Шаблон ControlTemplate имеет только один элемент управления в качестве корневого элемента. При этом корневой элемент управления содержит другие элементы управления.

В программном коде показаны шаблоны OrangeTemplate и GreenTemplate. Корневым элементом каждого из шаблонов является элемент

управления Grid. Комбинация элементов управления внутри корневого элемента управления составляет визуальную структуру шаблонной страницы.

Внутри корневого элемента находится три ячейки, располагающиеся друг под другом. Первая ячейка содержит заголовок страницы.

Первая ячейка имеет высоту 10% от высоты страницы и содержит элемент управления BoxView, заполняющий всю ячейку, а также элемент управления Label, в котором отображается название используемого шаблона. Для данных элементов управления произведена настройка расположения, цвета заполнения, а также цвета и размеров шрифта, а также отступов от края элемента управления.

Вторая ячейка содержит элемент управления ContentPresenter, в котором будут размещаться элементы управления, располагающиеся в странице, которая обращается к шаблону. Вторая ячейка имеет высоту 80% от высоты страницы.

Третья ячейка представляет собой колонтитул страницы, имеет высоту 10% от высоты шаблонной страницы и содержит элемент управления BoxView, заполняющий всю ячейку, а также элемент управления Label, на который необходимо нажать, если необходимо переключить используемый шаблон. Для данных элементов управления также произведена настройка расположения, цвета заполнения, а также цвета и размеров шрифта, стиля написания текста, а также отступов от края элемента управления.

Расширение разметки TemplateBinding с помощи свойства HeaderText разрешает передачу значения свойству Text для шаблонного элемента управления Label из программного кода в файле MainPage.xaml.

Также необходимо отметить, что к элементу управления Label, расположенному в нижней части страницы, привязан обработчик события к OnChangeTheme, предназначенный для обработки жеста касания и приведенный в файле HeaderFooterPage.xaml.cs, который будет рассмотрен далее. Также элемент управления связан с процедурой OnApplyTemplate, отвечающей за смену шаблона отображения страницы и приведенной в файле MainPage.xaml.cs.

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
        xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        xmlns:controls="clr-namespace: TemPage"
        x:Class="TemPage.HeaderFooterPage">
  <ContentPage.Resources>
    <Style TargetType="Entry">
      <Setter Property="VisualStateManager.VisualStateGroups">
        <VisualStateGroupList>
          <VisualStateGroup x:Name="CommonStates">
            <VisualState x:Name="Normal">
              <VisualState.Setters>
                <Setter Property="FontSize"
                    Value="18" />
              </VisualState.Setters>
```

```
            </VisualState>
            <VisualState x:Name="Focused">
              <VisualState.Setters>
                <Setter Property="FontSize"
                        Value="36" />
              </VisualState.Setters>
            </VisualState>
          </VisualStateGroup>
        </VisualStateGroupList>
      </Setter>
    </Style>

    <ControlTemplate x:Key="OrangeTemplate">
      <Grid>
        <Grid.RowDefinitions>
          <RowDefinition Height="0.1*" />
          <RowDefinition Height="0.8*" />
          <RowDefinition Height="0.1*" />
        </Grid.RowDefinitions>
        <BoxView Color="Orange" />
        <Label Margin="20,0,0,0"
               Text="{TemplateBinding HeaderText}"
               TextColor="White"
               FontSize="20"
               HorizontalOptions="Center"
               VerticalOptions="Center" />
        <ContentPresenter Grid.Row="1" />
        <BoxView Grid.Row="2"
                 Color="Orange" />
        <Label x:Name="ChangeThemeLabel"
               Grid.Row="2"
               Margin="20,0,0,0"
               Text="Change Theme"
               TextColor="White"
               FontSize="20"
               HorizontalOptions="Center"
               VerticalOptions="Center">
          <Label.GestureRecognizers>
            <TapGestureRecognizer
              Tapped="OnChangeTheme" />
          </Label.GestureRecognizers>
        </Label>
      </Grid>
    </ControlTemplate>

    <ControlTemplate x:Key="GreenTemplate">
      <Grid>
        <Grid.RowDefinitions>
          <RowDefinition Height="0.1*" />
```

```
            <RowDefinition Height="0.8*" />
            <RowDefinition Height="0.1*" />
          </Grid.RowDefinitions>
          <BoxView Color="Green" />
          <Label Margin="20,0,0,0"
              Text="{TemplateBinding HeaderText}"
              TextColor="Yellow"
              FontSize="20"
              HorizontalOptions="Center"
              VerticalOptions="Center" />
          <ContentPresenter Grid.Row="1" />
          <BoxView Grid.Row="2"
               Color="Green" />
          <Label x:Name="ChangeThemeLabel"
              Grid.Row="2"
              Margin="20,0,0,0"
              Text="Change Theme"
              FontSize="20" FontAttributes="Bold"
              TextColor="Yellow"
              HorizontalOptions="Center"
              VerticalOptions="Center">
            <Label.GestureRecognizers>
              <TapGestureRecognizer
                Tapped="OnChangeTheme" />
            </Label.GestureRecognizers>
          </Label>
        </Grid>
      </ControlTemplate>
    </ContentPage.Resources>
</ContentPage>
```

Откроем файл HeaderFooterPage.xaml.cs. Наберем в файле программный код в соответствии с презентацией.

В программном коде сначала объявляются рассмотренные ранее шаблоны OrangeTemplate и GreenTemplate.

Затем объявляется свойство HeaderTextProperty для записи значения шаблонного свойства HeaderText. Производится инициализация страницы HeaderFooterPage типа ContentPage.

Для отслеживания переключения между шаблонами OrangeTemplate и GreenTemplate служит свойство OriginalTemplate и вспомогательная переменная original.

Для обработки жеста нажатия на элемент управления Label, предназначенной для смены шаблона страницы, предназначен обработчик OnChangeTheme, на который ссылается программный код xaml страницы HeaderFooterPage.

```
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
```

```
namespace TemPage
{
  [XamlCompilation(XamlCompilationOptions.Compile)]
  public partial class HeaderFooterPage : ContentPage
  {
    ControlTemplate OrangeTemplate;
    ControlTemplate GreenTemplate;

    public static readonly BindableProperty
      HeaderTextProperty =
      BindableProperty.Create(nameof(HeaderText),
                  typeof(string),
                  typeof(HeaderFooterPage),
                  default(string));
    public string HeaderText
    {
      get => (string)GetValue(HeaderTextProperty);
      set => SetValue(HeaderTextProperty, value);
    }
    bool original = true;
    public bool OriginalTemplate
    {
      get { return original; }
    }

    public HeaderFooterPage()
    {
      InitializeComponent();
      OrangeTemplate =
        (ControlTemplate)Resources["OrangeTemplate"];
      GreenTemplate =
        (ControlTemplate)Resources["GreenTemplate"];
    }
    void OnChangeTheme(object sender, EventArgs e)
    {
      original = !original;
      ControlTemplate =
        (original) ? OrangeTemplate : GreenTemplate;
    }
  }
}
```

Откроем файл MainPage.xaml и наберем программный код в соответствии с презентацией.

Следует отметить, что в программном коде производится ссылка на страницу HeaderFooterPage, в которой хранятся шаблоны, а также настройки для элементов управления Entry.

Свойство HeaderText предназначено для передачи текстовой строки «Страница TemplatedPage» для значения свойства Text элемента управления Label, расположенного в заголовке шаблонной страницы.

Страница MainPage, к которой применяется шаблон, передает в элемент управления ContentPresenter шаблонной страницы макет типа StackLayout с двумя элементами управления Entry и элементом управления Button. К элементам управления Entry применяются стандартные настройки для визуального отображения, приведенные в программном коде страницы HeaderFooterPage.

```
<controls:HeaderFooterPage xmlns="http://xamarin.com/schemas/2014/forms"
        xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        xmlns:controls="clr-namespace:TemPage"
        x:Class="TemPage.MainPage"
        Title="Access template"
        HeaderText="Страница TemplatedPage"
        ControlTemplate="{StaticResource OrangeTemplate}">
  <StackLayout Margin="10">
    <Entry Placeholder="Enter username" />
    <Entry Placeholder="Enter password"
          IsPassword="True" />
    <Button Text="Login" />
  </StackLayout>
</controls:HeaderFooterPage>
```

Откроем файл MainPage.xaml.cs и наберем программный код в соответствии с презентацией.

Переменная TemLabel используется для связи с элементом управления Label, расположенным в колонтитуле шаблонной страницы и имеющим в соответствии с атрибутом x:Name имя ChangeThemeLabel.

Процедура OnApplyTemplate предназначена для изменения текста в элементе управления Label в шаблонной странице в случае нажатия на этот элемент управления при изменении шаблона отображения страницы.

```
using Xamarin.Forms;
namespace TemPage
{
  public partial class MainPage : HeaderFooterPage
  {
    Label TemLabel;
    public MainPage()
    {
      InitializeComponent();
    }
    protected override void OnApplyTemplate()
    {
      base.OnApplyTemplate();
```

```
            TemLabel = (Label)GetTemplateChild("ChangeThemeLabel");
            TemLabel.Text =
                OriginalTemplate ? "Orange Theme" : "Green Theme";
        }
    }
}
```

Совершим сборку программного приложения, после которой не должно быть ошибок, и запустим программное приложение.

Сначала отображается страница в соответствии с шаблоном OrangeTemplate.

Внутри страницы располагаются два элемента управления Entry, а также элемент управления Button, которые в составе макета StackLayout передаются шаблонному элементу управления ContentPresenter.

При нажатии на элемент управления Label в колонтитуле, в котором отображено название используемого шаблона, производится смена шаблона отображения страницы MainPage.

В случае начала работы с элементами управления Entry они меняют свое визуальное состояние (изменяется размер шрифта).

После окончания работы с элементами управления Entry они возвращаются в исходное состояние.