DOI 10.34185/1562-9945-1-144-2023-07
UDC 004.4

I.V. Ponomarev

FEATURES OF THE .NET MAUI FRAMEWORK FOR CREATING A CROSS-PLATFORM APPLICATIONS

Abstract. Modern technologies that allow you to create applications for several different platforms optimize the development process. The recently released .NET MAUI platform is a new milestone in the development of cross-platform development technology. The possibilities of the platform provided to the programmer and the features of creating applications are considered.
Keywords: .NET, MAUI, CROSS-PLATFORM, C#, XAML, Android, iOS, macOS, Windows, Tizen, Page, Layout, View.

Formulation of the problem. Developing an application for several platforms at once (iOS, Android, Windows, macOS) faces many difficulties. To create an application for a specific platform, the appropriate tools, development environments and programming languages are used. The development is also influenced by the difference in approaches to building a graphical interface and its specific features. The presence of a wide range of platforms, development tools and programming languages increases both the time and cost of development.

Cross-platform provides the ability to easily create applications for multiple platforms. The Xamarin platform allows you to write cross-platform applications using the C# programming language in the modern Visual Studio environment. With Xamarin, it is possible to create a custom user interface for each platform and write common business logic in C# that will be used on different platforms. Xamarin offers support for three main platforms: iOS, Android, and Windows.

The Xamarin.Forms UI framework allows developers to create user interfaces in the XAML markup language using C# code-behind. These user interfaces are rendered as native controls on each platform.

Thus, the development costs and time of release of mobile products are significantly reduced. Further improvement of this technology led to the creation of a new platform .NET MAUI.

Purpose of the research. Необходимо рассмотреть возможности платформы .NET MAUI, ее преимущества перед предыдущей платформой, а также, особенности создания на этой системе кроссплатформенных приложений.

Main part. The .NET Multiplatform Application User Interface (.NET MAUI) is a cross-platform framework for building native mobile and desktop applications using the C# programming language and XAML. The developed applications can run on Android, iOS, macOS, Windows and Tizen from a single common code base. .NET MAUI is an evolution of Xamarin.Forms extended from mobile to desktop scenarios, with UI controls redesigned for performance and extensibility.

Benefits of developing applications on MAUI:

- in the process of development, a single project is created that uses a common code for all platforms;
- when creating applications, the .NET platform and the C# programming language are used;
- offers a rich collection of built-in controls;
- data binding is supported;
- there are options for customizing the behavior of the visual interface and built-in functionality;
- wide opportunities for working with graphics;
- providing direct access to the native API of each platform;
- .NET hot reload simplifies development.

A single project enables simplified and consistent cross-platform development regardless of target platforms and provides the following features:

- one shared project designed for Android, iOS, macOS, Windows and Tizen;
- simplified debug target selection for launching applications;
- shared project resource files;
- if necessary, access to API and tools for specific platforms;
- a single entry point for a cross-platform application [1].

.NET MAUI provides a large set of controls that you can use to display data, initiate actions, indicate activity, display collections, select data, and more.

Support for .NET hot reloading allows you to change managed source code and XAML files while your application is running and see the changes without recompiling.

The .NET MAUI project separates the GUI definition from the program logic by default. Extensible Application Markup Language XAML is used to define the user interface. XAML is an XML-based markup language for creating objects in a

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

declarative way. While it is also possible to create an interface in C# code, or combine interface creation in XAML and C# code, XAML is recommended because it is often more concise, visually consistent, and has tooling support. XAML allows you to organize your entire user interface into a set of pages, much like HTML does.

Each element in XAML represents an object of a particular C# class, and the element's attributes map to properties of those classes. When you add a new XAML page to your project, a C# code file is also added at the same time. So, when you create a project, by default, a file with a graphical interface in XAML is added to it - MainPage.xaml and a MainPage.xaml.cs file, where the application logic related to the markup from the XAML file should be located. XAML files let you define the window's interface, but you use C# code to create application logic, such as defining control event handlers.

By default, the x:Class attribute is defined in the XAML file:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns=http://schemas.microsoft.com/dotnet/2021/maui
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="App.MainPage">
   <ScrollView>
       <VerticalStackLayout
           Spacing="25"
           Padding="30,0"
           VerticalOptions="Center">
     …
           <Button
               x:Name="CounterBtn"
               Text="Click me"
               SemanticProperties.Hint="Counter"
               Clicked="OnCounterClicked"
               HorizontalOptions="Center" />
       </VerticalStackLayout>
    </ScrollView>
</ContentPage>
```

The x:Class attribute points to the class that will represent this page and into which code from XAML will be compiled.

The corresponding code file MainPage.xaml.cs has the following code:

```
namespace App;
public partial class MainPage: ContentPage
{
    int count = 0;
    public MainPage()
```

```
        {
            InitializeComponent();
        }
        private void OnCounterClicked(object sender, EventArgs e)
        {
            count++;
            CounterBtn.Text = $"Clicked {count} time(s)";
            SemanticScreenReader.Announce(CounterBtn.Text);
        }
    }
```

The class constructor retrieves the GUI code, initializes all the objects defined in the XAML file, sets the element structure, attaches the event handlers defined in the C# code file, and creates the object tree. So the code from XAML is used to build the interface.

In the XAML definition of the button, the x:Name attribute assigns a name to the element. When compiling the application, a private variable with this name will be created. The Clicked attribute sets the button's click handler. It has the value "OnCounterClicked" and the C# code has a method with the given name that handles clicks on this button.

The user interface of a .NET MAUI application consists of objects that map to the native controls of each target platform. The main control groups used to create the user interface of an application are Pages, Layouts, and Views.

A .NET MAUI page usually takes up the entire screen or window. It contains a layout which contains views and possibly other layouts.

.NET MAUI defines the following pages:

• ContentPage displays a single view and is the most common page type.

• NavigationPage provides hierarchical navigation, allowing you to move back and forth through pages as you wish.

• FlyoutPage manages two related information pages: a flyout page with items, and a detail page that provides details about the items on the flyout page. Different behaviors apply for phones and for tablets or desktops.

• A TabbedPage consists of a series of pages that can be navigated through using tabs at the top or bottom of the page, and each tab loads the content of the page [2].

.NET MAUI layouts are used to group UI controls into visual structures, and each layout typically contains multiple views. Layout classes usually contain logic for setting the position and size of child elements.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

.NET MAUI has the following layouts:

• StackLayout arranges nested elements in a row either horizontally or vertically.

• AbsoluteLayout allows you to set the exact size and position coordinates for nested elements on the page.

• Grid arranges nested elements in the form of a table.

• FlexLayout allows you to lay out or wrap your child elements with different alignment and orientation options [2].

.NET MAUI views are user interface objects such as labels, buttons, and sliders, commonly referred to as controls in other environments. For all views it is possible to set the height and width, outer and inner padding, horizontal and vertical alignment.

Resources are used to share different elements of common components. Resources include styles, control templates, data templates, converters, and colors. Any visual object can reference XAML resources.

Data binding is one of the key aspects when building applications on the .NET MAUI platform. It allows you to link two properties of different objects in such a way that changes in the property of one object automatically lead to a change in the property of another object. Data binding in XAML helps reduce the size of your code-behind file. By replacing procedural code in event handlers with declarative code or markup, the application is simplified and more understandable.

Triggers in .NET MAUI also allow you to declaratively specify some actions that are performed when properties of a visual element change, data changes, or events are fired.

To create a .NET MAUI application that uses cross-platform code, follow these steps.

1. When creating a project, you must select a debug target (for example, Android Emulator).

2. Specify the user interface using XAML and interact with XAML elements using C# code.

3. Views are created and their data binding.

4. Navigation is used to move through the pages of the application.

Conclusions. The .NET MAUI platform provides the ability to create both mobile and desktop applications with a single interface. The developer is offered ample opportunities to organize the structure and select user interface controls. Separating the GUI definition from the program logic allows for more efficient

development and maintenance of applications. In addition, MAUI's deep integration with other .NET tools and services ensures high performance in the applications you create.

REFERENCES

1. What is .NET MAUI? – .NET Multi-platform App UI documentation | Microsoft Docs [Electronic resource]. Access mode: https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui?view=net-maui-7.0

2. Controls – .NET Multi-platform App UI documentation | Microsoft Docs [Electronic resource]. Access mode: https://learn.microsoft.com/en-us/dotnet/maui/user-interface/controls/ ?view=net-maui-6.0

Received 16.01.2023.
Accepted 18.01.2023.

Можливості фреймворку .NET MAUI зі створення кросплатформових додатків

Кросплатформеність надає можливість легко і просто створювати програми для декількох платформ. Широко використовувана платформа Xamarin дозволяє створювати власний інтерфейс користувача для кожної платформи і писати мовою програмування C# загальну бізнес-логіку, яка буде використовуватися на різних платформах. Xamarin пропонує підтримку для трьох основних платформ: iOS, Android та Windows. Однак існуюча платформа Xamarin має ряд недоліків. Подальше вдосконалення даної технології спричинило створення нової платформи .NET MAUI.

Необхідно розглянути можливості платформи .NET MAUI, її переваги перед попередньою платформою, а також особливості створення на цій системі кросплатформових додатків.

Інтерфейс багатоплатформної програми .NET — це кросплатформовий фреймворк для створення власних мобільних і настільних програм за допомогою мови програмування C# і XAML. Розроблені програми можуть працювати на Android, iOS, macOS, Windows та Tizen з єдиної загальної бази коду. У процесі розробки додатків на MAUI створюється єдиний проект, що використовує загальний код для всіх платформ, використовується платформа .NET та мова програмування C#. Розробнику надається багата колекція вбудованих елементів управління, можливості прив'язки даних, налаштування поведінки візуального інтерфейсу та вбудованого функціоналу, широкі можливості роботи з графікою. При необхідності існує прямий доступ до нативних API кожної платформи та гаряче перезавантаження .NET, яке спрощує розробку та налагодження програми.

Таким чином, платформа .NET MAUI надає можливість створення як мобільних, так і настільних додатків з єдиним інтерфейсом. Розробнику

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

пропонуються широкі можливості з організації структури та вибору елементів управління інтерфейсу користувача. Поділ визначення графічного інтерфейсу від програмної логіки дозволяє більш ефективно розробляти та підтримувати програми. Крім того, глибока інтеграція MAUI з іншими інструментами та службами .NET забезпечує високу продуктивність створюваних додатків.

Пономарьов Ігор Володимирович - кандидат технічних наук, доцент кафедри електронних обчислювальних машин факультету фізики, електроніки та комп'ютерних систем Дніпровського національного університету ім.О.Гончара.

Ponomarev Igor Volodimirovich - candidate of technical sciences, associate professor of the department of electronic computers of the faculty of physics electronics and computer systems ofthe Oles Honchar Dnipro National University.