Л9. Крос-платформна розробка. Технологія Xamarin

Xamarin.Forms представляє платформу, яка націлена на створення кросплатформових додатків під Android, iOS та Windows 10. Навіщо використовувати саме цю платформу, які переваги вона несе? Є певні статистичні дані, що значна частина мобільних програм створюється більш ніж для однієї платформи, наприклад, для Android та iOS. Однак неминуче розробники стикаються з такими труднощами:

- ***відмінність у підходах побудова графічного інтерфейсу*** *впливає розробку. Розробники змушені підлаштовувати програму під вимоги до інтерфейсу на конкретній платформі;*
- ***різні API*** *- відмінність у програмних інтерфейсах та реалізаціях тих чи інших функціональностей також вимагає від програміста облік цих специфічних особливостей;*
- ***різні платформи для розробки****. Наприклад, щоб створювати програми для iOS необхідне відповідне середовище – macOS і ряд спеціальних інструментів типу XCode, як мову програмування вибирається Objective-C або Swift; для Android використовуються середовища - Android Studio, Eclipse , як мова програмування застосовується Java або Kotlin; для створення програм під Windows використовується Visual Studio та мови - C#, F#, VB.NET, C++.*

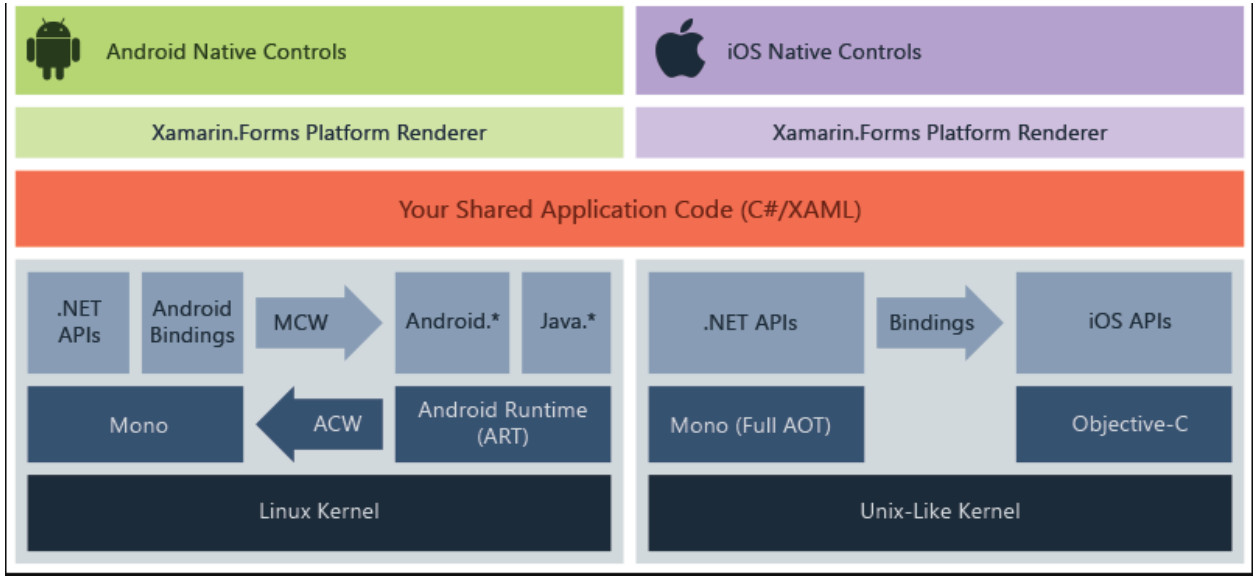
Такий діапазон платформ, засобів розробки та мов програмування не може позитивно позначатися на термінах створення додатків, і, зрештою, на коштах, що виділяються на розробку. Було б дуже ефективно мати один інструмент, який дозволяв легко і просто створювати програми відразу для всіх платформ. Таким інструментом є платформа Xamarin (вимовляється як "земарин").

Л9. Крос-платформна розробка. Технологія Xamarin

Xamarin дозволяє створювати одну єдину логіку програми із застосуванням C# і .NET відразу для всіх трьох платформ – Android, iOS, UWP. Переваги використання Xamarin.Forms:

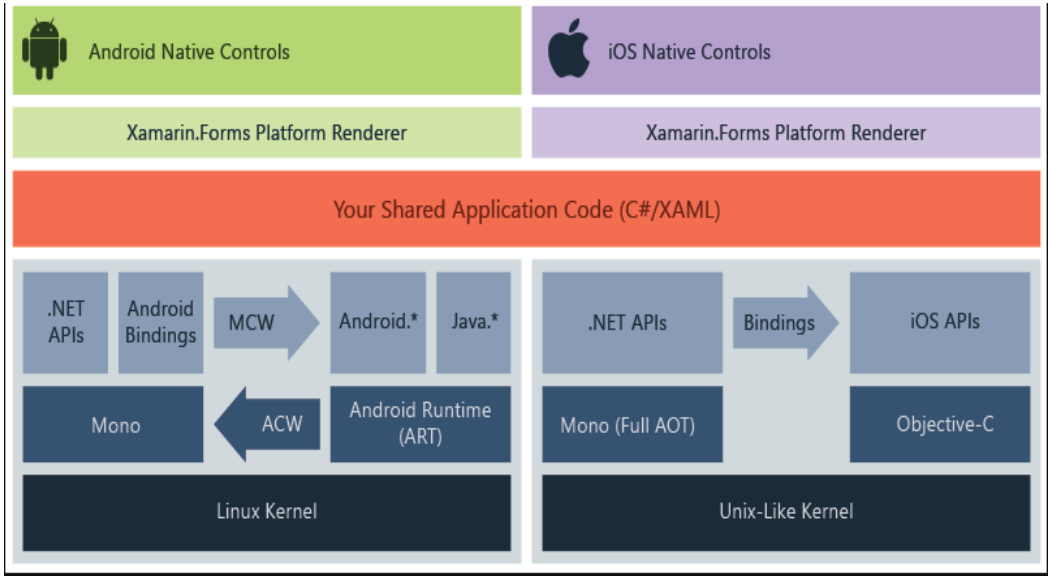
- *у процесі розробки створюється єдиний код для всіх платформ;*
- *Xamarin надає прямий доступ до нативних API кожної платформи;*
- *Xamarin Forms підтримує декілька платформ: Android, iOS, UWP, Tizen.*

*Xamarin працює поверх фреймворку **Mono**, який надає opensource-реалізацію .NET Framework. Mono може працювати поверх різних платформ – Linix, MacOS тощо. На рівні кожної окремої платформи Xamarin покладається на низку субплатформ. Зокрема: Xamarin.Android - бібліотеки для створення програм на ОС Android. Xamarin.iOS - бібліотеки для створення програм для iOS. Ці субплатформи відіграють велику роль - через них програми можуть надавати запити до прикладних інтерфейсів на пристроях під керуванням ОС Android або iOS.*



Л9. Крос-платформна розробка. Технологія Xamarin

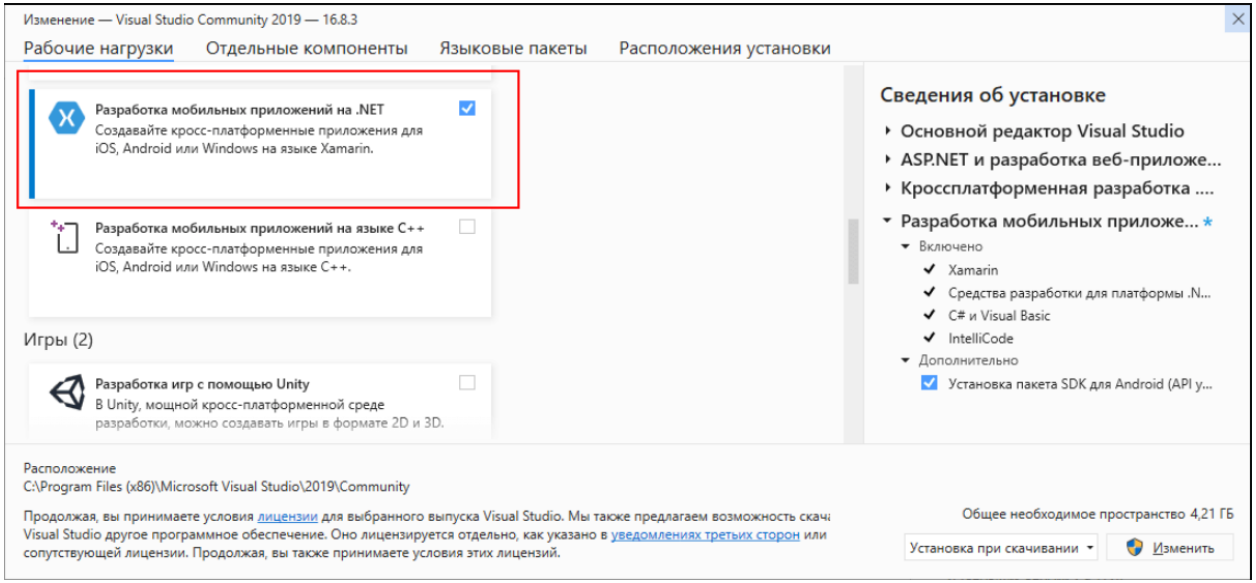
За допомогою Xamarin.Android код C# з використанням Xamarin компілюється ***в Intermediate Language (IL)****, який після запуску програми компілюється в нативну збірку. Xamarin-програми запускаються в середовищі виконання* ***Mono****. Безпосередньо код не може звертатися до API Android. Для цього треба звернутися до функціональності просторів імен* ***Android.* і Java.****, які надаються віртуальною машиною Android Runtime (ART).*



Спеціальний прошарок ***Managed Callable Wrappers (MCW)*** *дозволяє транслювати виклики managed-коду в нативні виклики та звертатися до функціональності просторів імен Android.* та Java.* І навпаки, коли Android Runtime (ART) звертається до програми з кодом Xamarin, всі виклики проходять через обгортку* ***Android Callable Wrappers (ACW)****. Програми Xamarin.iOS на відміну від Xamarin.Android, який використовує JIT-компіляцію, застосовують* ***AOT-компіляцію*** *(Ahead-of-Time) коду C# в нативний ARM-код. Xamarin використовує проміжний шар Selectors (селектори) для трансляції викликів коду Objective-C код на C# і шар Registrars (реєстратори) для трансляції коду C# Objective-C. У результаті шари Selectors і Registrars загалом представляють переміжний шар, який на ілюстрації вище позначений як "****bindings****" і який дозволяє взаємодіяти коду Objective-C з кодом C#.*

Л9. Крос-платформна розробка. Технологія Xamarin

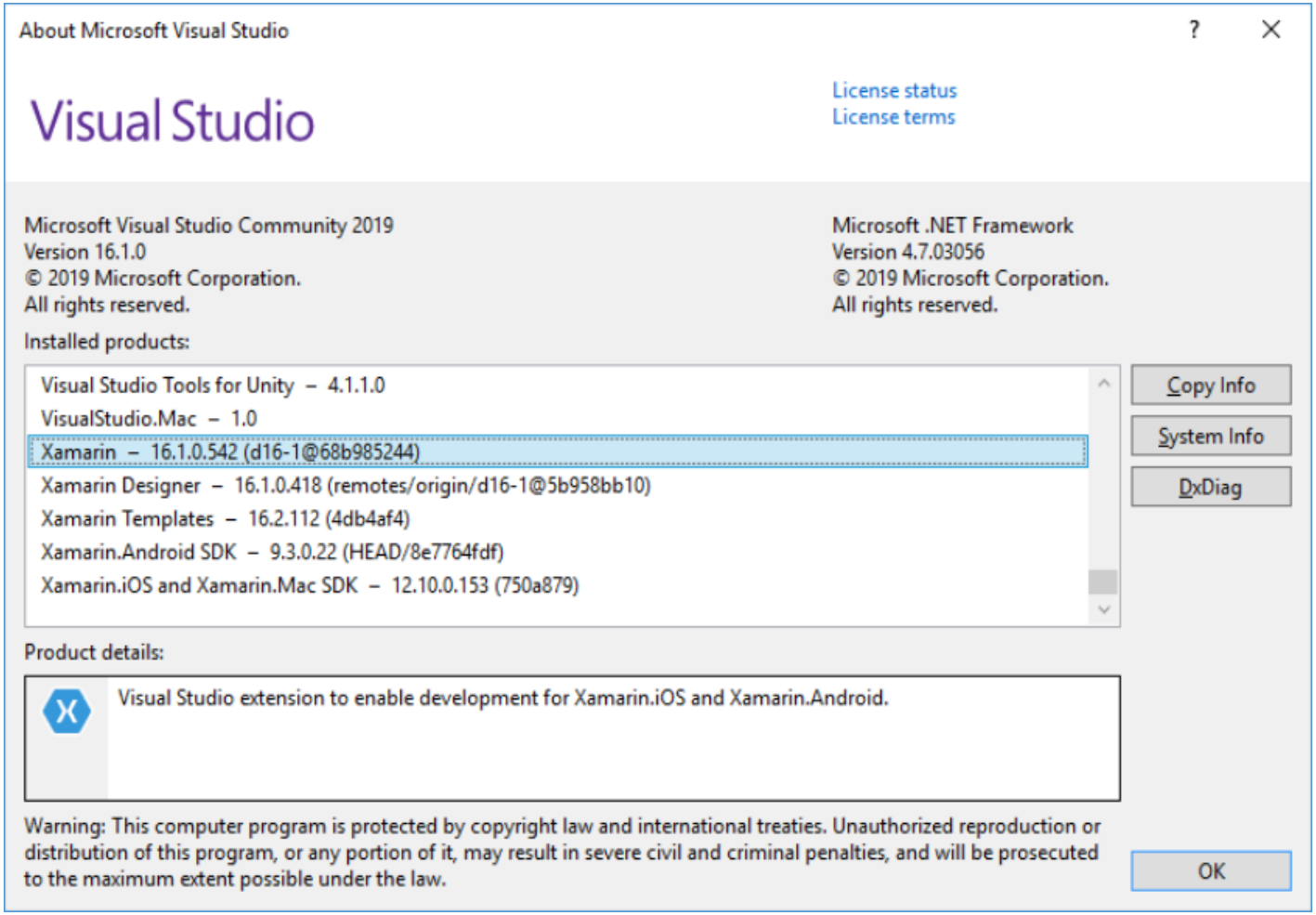
Найбільш важливою особливістю Xamarin є можливість створювати ***кросплатформові додатки*** *– тобто одна логіка для всіх платформ. Дана можливість представлена технологією* ***Xamarin.Forms*** *і яка працює рівнем вище Xamarin.Android і Xamarin.iOS. Тобто за допомогою Xamarin.Forms один раз можемо визначити візуальний інтерфейс, один раз до нього прив'язати якусь логіку на C# і все це буде працювати на Android, iOS та Windows. Потім Xamarin.Forms за допомогою рендererів (renderer) – спеціальних об'єктів для зв'язку контролів на XAML/C# з нативними контролами транслюють візуальні компоненти Xamarin.Forms у графічний інтерфейс, специфічний для кожної платформи.*



Для розробки кросплатформових додатків на Xamarin необхідно встановити середовище ***Visual Studio 2019 Community****. При встановленні Visual Studio 2019 у програмі для установника обов'язково треба вибрати пункт "Розробка мобільних програм на .NET"*

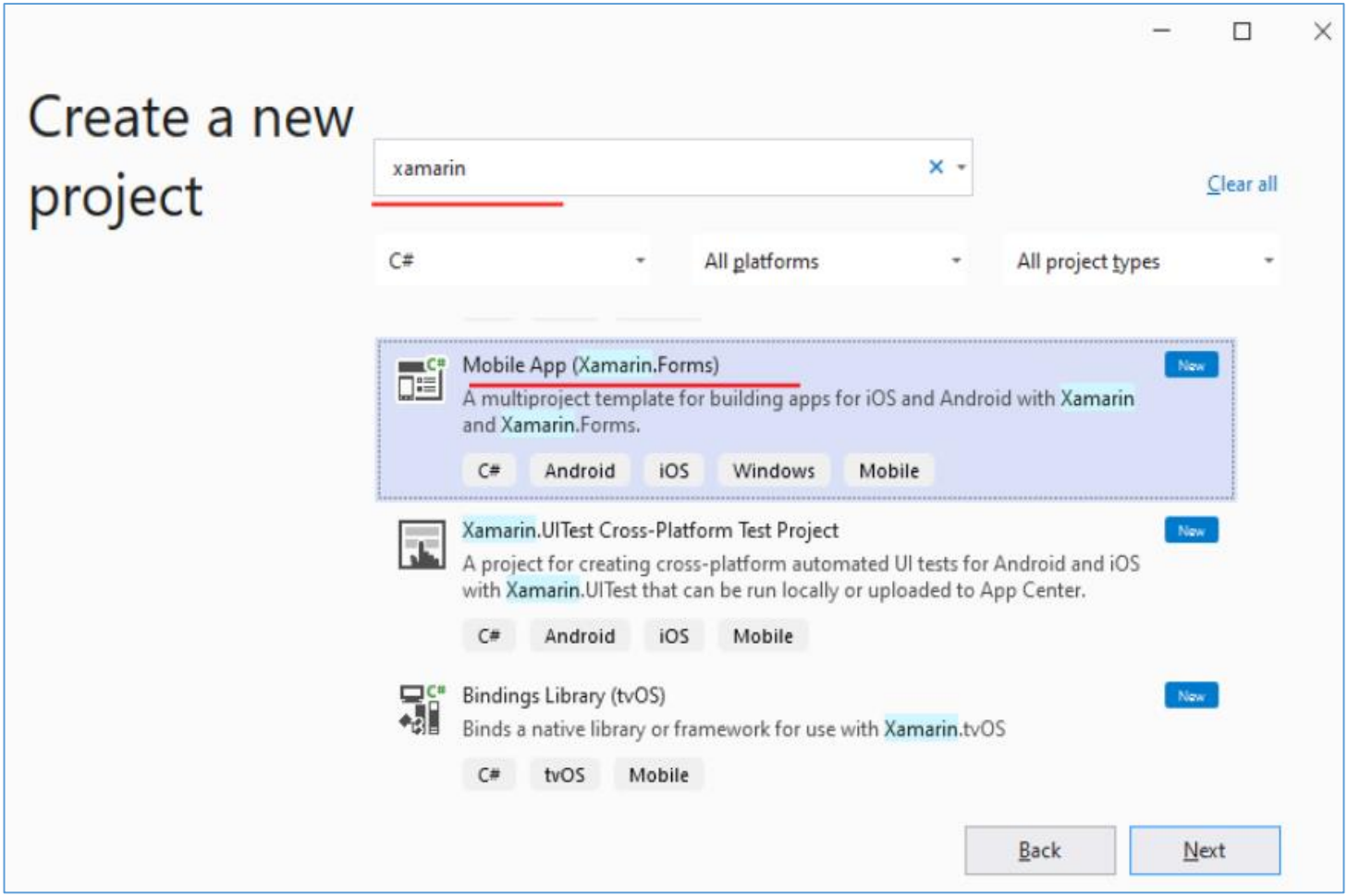
Установка Xamarin

Після інсталяції перевіряємо, що Xamarin встановлений. Для цього переходимо в меню Help -> About Microsoft Visual Studio (рис. 2). Якщо цільовою операційною системою є macOS X, то необхідно встановити Visual Studio for Mac. Крім того, для розробки на macOS X потрібно встановити XCode, доступний в AppStore



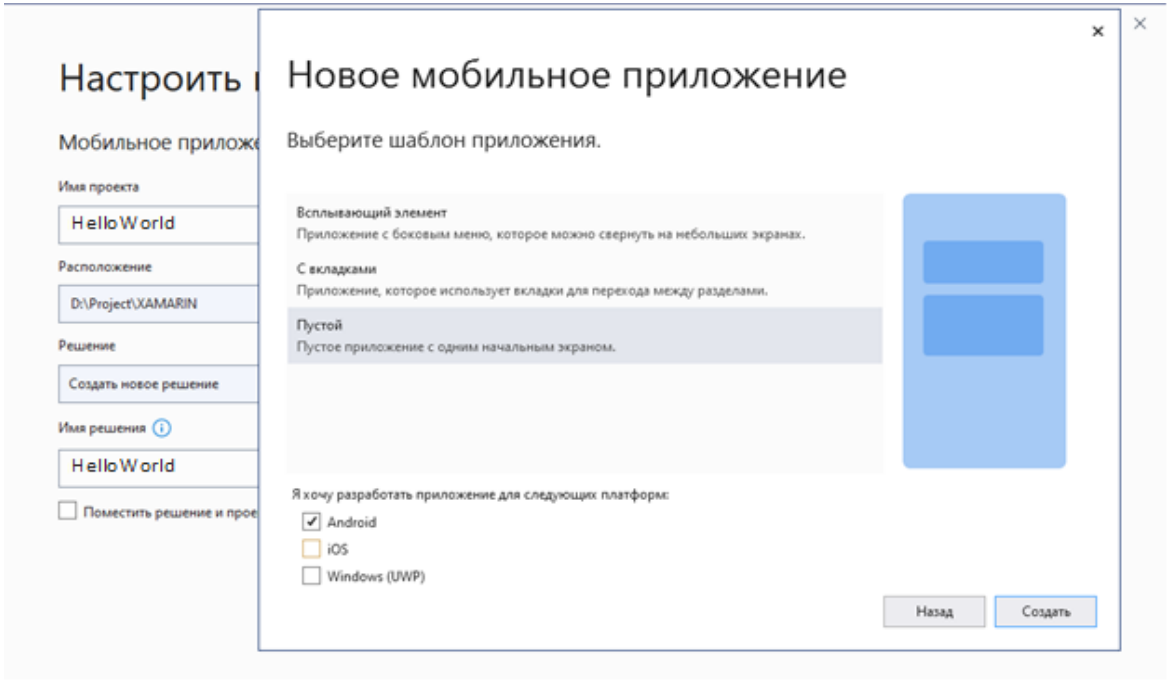
Створення та запуск проекту

Для створення кросплатформних програм для Xamarin Forms у Visual Studio 2019 призначений шаблон проекту, який називається Mobile App (Xamarin.Forms)



Створення та запуск проекту

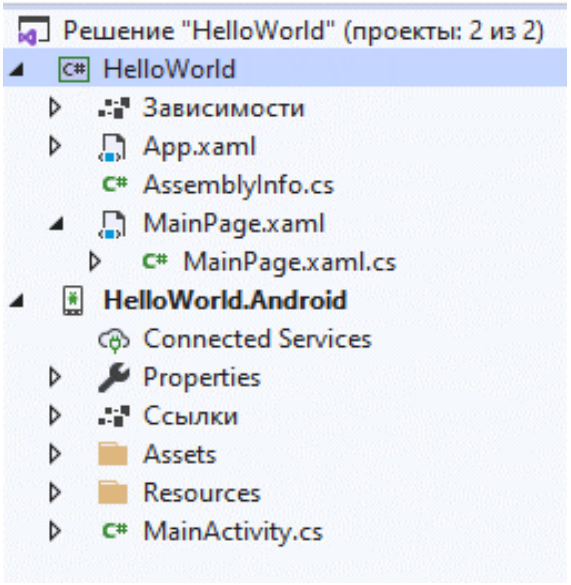
*Створюємо проект HelloWorld. Шаблони **Floyout** – проект з бічним меню, **Tabbed** – проект програми, яка використовує вкладки для навігації між сторінками, **Blank** –порожній шаблон, що створює проект з мінімальною функціональністю. Вибираємо шаблон Blank*



При створенні проекту в опції Platform вибираємо під які ОС буде створюватись проект: Android, iOS, Windows (UWP). Windows (UWP) доступний під Windows, якщо для Visual Studio встановлено інструмент для розробки під UWP.

Створення та запуск проекту

До рішення входить єдиний проект, який назвали HelloWorld. Його структура представлена на рис.



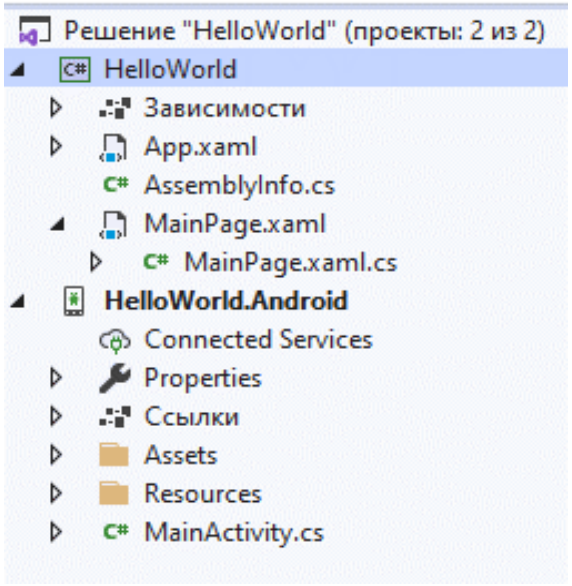
У головному проекті буде чотири основні файли:
***App.xaml**: файл, який визначає ресурси, спільні для всієї програми;*
***App.xaml.cs**: файл із кодом C#, з якого починається виконання програми;*
***MainPage.xaml**: файл з візуальним інтерфейсом для єдиної сторінки MainPage у вигляді xaml;*
***MainPage.xaml.cs**: файл, який містить логіку MainPage мовою C#;*
***AssemblyInfo.cs**: файл із кодом мовою C#, який використовується для встановлення налаштувань програм*

До проекту входять такі файли та каталоги, що забезпечують побудову програми з мінімальною функціональністю.

***Connected Service**s – призначений для підключення різноманітних хмарних сервісів, які використовуються у проекті. За замовчуванням нічого не містить, але при підключенні хмарної служби Visual Studio додасть необхідні посилання, код підключення та внесе зміни до певних файлів конфігурації.*

***Properties** – ця папка містить файл маніфесту AndroidManifest.xml. У ньому визначені описи всіх вимог, необхідних для Xamarin.Android, зокрема вказані його ім'я, номер версії та дозволи, що використовуються. Крім того, в папці Properties ще розміщується файл **AssemblyInfo.cs**, в якому вказуються метадані зборки .NET. У нього вносять базові відомості про програму*

Створення та запуск проекту



Посилання *– ця папка включає зборки, необхідні для розробки та запуску програми. Вибравши, а потім розгорнувши каталог Посилання (References), можна побачити посилання на бібліотеки .NET, зокрема System, System.Core, System.Xml, посилання на зборку Xamarin - Mono.Android. У цей каталог у процесі роботи з проектом додаються інші бібліотеки, необхідні для розробки.*

MainActivity.cs *– файл C #, що містить опис класу головного стартового екрана програми.*

Assets *– до цієї папки включаються файли, необхідні для виконання програми, такі як шрифти, локальні файли даних та текстові файли. Доступ до файлів у цьому каталозі здійснюється за допомогою створюваного екземпляра диспетчера AssetManager.*

Resources *– це каталог, який містить ресурси програми. До таких відносяться рядки, малюнки та макети. Для того щоб звертатися до ресурсів у коді, створюється клас Resource, за допомогою якого відбувається отримання ресурсів, що використовуються в коді. Він також знаходиться в цій папці.*

Ресурси програми

*Каталог **Resources** включає ряд підкаталогів, таких як **drawable**, **layout**, **mipmap** та **values**. Крім того, він містить файл Resource.designer.cs. Призначення цих елементів описано нижче:*

***drawable** – призначений для розміщування графічних ресурсів, наприклад зображень, зокрема растрових. Доступ до ресурсів у програмному коді цього каталогу можливий за допомогою класу Resource.Drawable;*

***mipmap** – каталог, що містить файли, що промальовуються, для значків запуску з різною щільністю. Доступ до ресурсів цього каталогу здійснюється з використанням класу Resource.Mipmap. Для створення таких значків можна використовувати online сервіс. https://romannurik.github.io/AndroidAssetStudio/icons-launcher.html.*

***layout** – папка, яка включає файли дизайнера Android (мають розширення axml або xml), які, у свою чергу, визначають візуальний інтерфейс для кожного екрану, фрагмента чи діяльності. У разі використання шаблону порожньої програми при створенні проекту буде сформовано файл компонування, який називається activity_main.xml. Для доступу до ресурсів цього каталогу використовується клас Resource.Layout.*

***values** – папка, що містить файли XML, призначені для зберігання простих значень, наприклад рядків, цілих чисел і кольорів. При створенні проекту з використанням шаблону порожньої програми, зокрема, будуть створені файли з ім'ям strings.xml для зберігання текстових значень та styles.xml для збереження стандартних стилів.*

Ресурси програми

Також можна додавати ресурси розмірів або кольорів. Для доступу до ресурсів з даного каталогу можна використовувати класи Resource.String для отримання рядків, Resource.Dimension для отримання розмірів, Resource.Color для визначення кольорів та ін.

У зв'язку з тим, що Android використовується на пристроях з різною роздільною здатністю екрана, потрібно мати в наявності растові зображення для декількох типів дозволів: низького, середнього, високого і дуже високого. Це дозволить забезпечити оптимальне поєднання якості зображень та продуктивності на всіх гаджетах. Такі можливості програми забезпечуються створенням альтернативних ресурсів, які мають зберігатися у додаткових вкладених каталогах у папці ***\Resources****. Поряд із папкою* ***drawable*** *вони створюються окремо.*

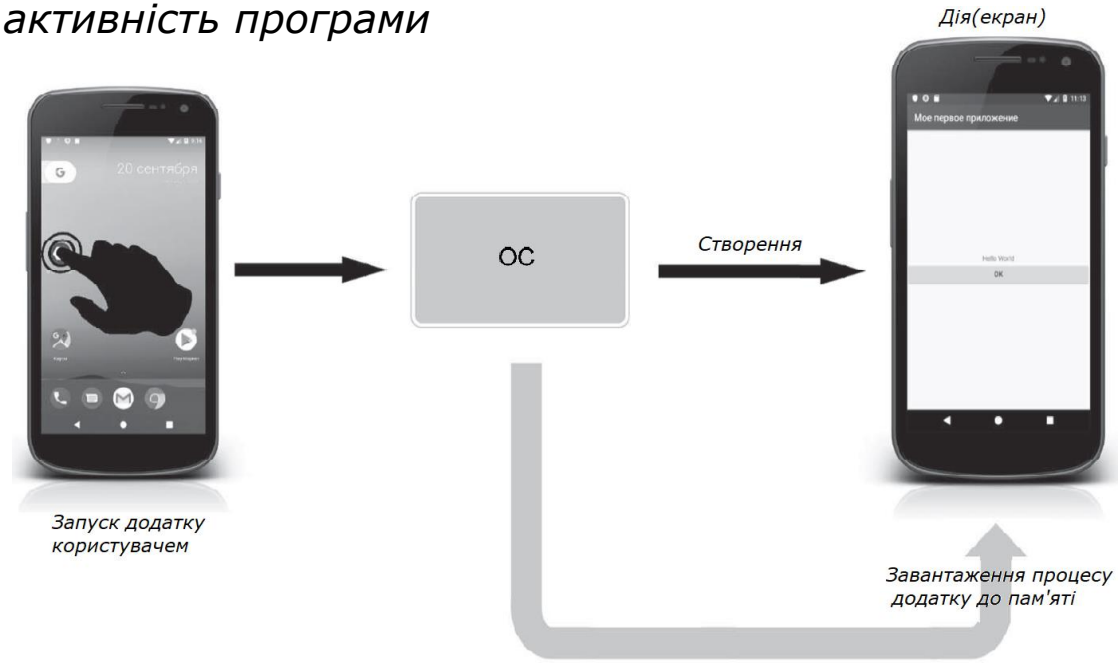
- ***mipmap-ldpi*** *– каталог, призначений для зберігання графічних ресурсів для екранів із низькою щільністю.*
- ***mipmap-mdpi*** *– каталог, призначений для зберігання графічних ресурсів для екранів із середньою щільністю.*
- ***mipmap-hdpi*** *– каталог, призначений для зберігання графічних ресурсів для екранів із високою щільністю.*
- ***mipmap-xhdpi*** *– каталог, призначений для зберігання графічних ресурсів для екранів з дуже високою густиною.*
- ***mipmap-xxhdpi*** *– каталог, призначений для зберігання графічних ресурсів для екранів з більш високою щільністю.*

Система автоматично зробить вибір відповідного залежно від роздільної здатності екрана пристрою, на якому виконується програма

Принципи роботи програми та основи архітектури

Програми для Android відрізняються відсутністю єдиної точки входу, тобто в додатку немає єдиного рядка коду, до якого операційна система звертається для його запуску. Запуск програми буде здійснено, коли система створює екземпляр одного із його класів. При цьому весь процес програми завантажується системою до пам'яті. Ця унікальна можливість Android має особливе значення, коли йдеться про створення складних програм або взаємодію з операційною системою Android.

Коли програма HelloWorld вперше відкривається на емуляторі або пристрої, Android створює першу дію (екран або активність). Останнє є собою особливий клас Android, який відповідає одному екранному уявленню та відповідає за відтворення та функціонування графічного інтерфейсу користувача. Процес додатку завантажується у момент, коли система створює першу активність програми



Принципи роботи програми та основи архітектури

*Оскільки в Android-додатку лінійна черговість виконання відсутня (програма може бути запущена з багатьох місць), система дозволяє відстежувати класи та файли, що входять до складу програми. У додатку HelloWorld реєстрація всіх частин програми відбувається у відповідному XML-файлі - **маніфесті Android**. Його призначення полягає у відстеженні вмісту, властивостей та дозволів програми. Він повідомляє про них безпосередньо операційну систему.*

Програма HelloWorld є фактично однією дією (екраном) та колекцією ресурсних та допоміжних пакетних файлів, пов'язаних маніфестом Android. Поряд з цим файл маніфесту відповідає за оголошення імені пакета програми, який є унікальним ідентифікатором, опис програмних компонентів, діяльності, служб, приймачів широкомовних повідомлень та контент-провайдерів, що дає можливість викликати класи реалізації кожного окремого компонента, а також оголошує їх наміри. У файл маніфесту включено список дозволів, необхідних для доступу до захищених частин API, а також для взаємодії з іншими програмами. Маніфест оголошує дозволи, які сторонні додатки повинні мати для взаємодії з компонентами цієї програми; оголошує мінімальний рівень API Android, необхідний роботи програми; перераховує зв'язані бібліотеки.

Створення інтерфейсу з коду

*Створемо проект **FirstApp** на базі шаблону **Blank**. Переходимо до файлу **MainPage.xaml**. Видаляємо усе крім ContentPage.*

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:local="clr-namespace:FirstApp"
             x:Class="FirstApp.MainPage">

</ContentPage>
```

*Переходимо до ф-лу **MainPage.xaml.cs**. Зміст файлу:*

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

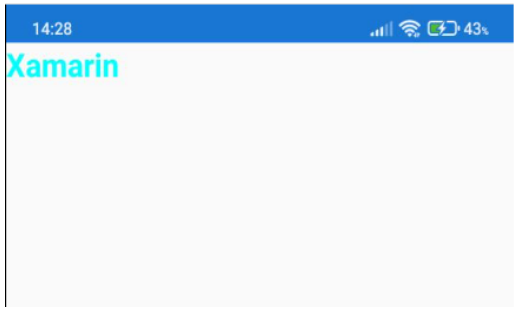
*Будемо створювати елементи в методі **OnAppearing(),** який є базовим батьківським методом ContentPage, тому його потрібно перевизначити.*
Створюємо макет типу StackLayout (стопка тарілок), який буде виступати батьківським елементом для всіх елементів, що будуть в середині.

```
protected override void OnAppearing()
{
    StackLayout layout = new Stacklayout();
}
```

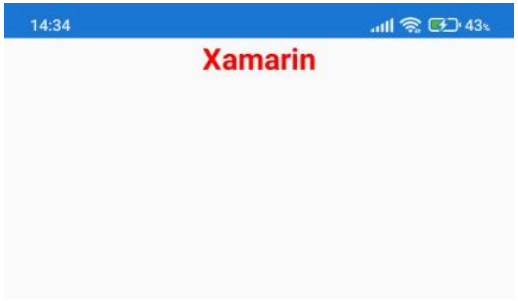
Створення інтерфейсу з коду

Додаємо Label та налаштуємо деякі властивості. На прикінці коду додаємо його до колекції елементів StackLayout.

```
protected override void OnAppearing()
{
    StackLayout layout = new StackLayout();

    Label label1 = new Label();
    label1.Text = "Xamarin";
    label1.TextColor = Color.Cyan;
    label1.FontSize = 20;
    label1.FontAttributes = FontAttributes.Bold;

    layout.Children.Add(label1);

    Content = layout;
}
```



Внесемо деякі правки. Додамо label1 центрування та змінимо колір на Red.

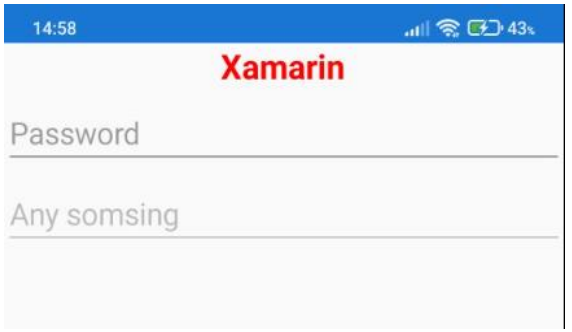


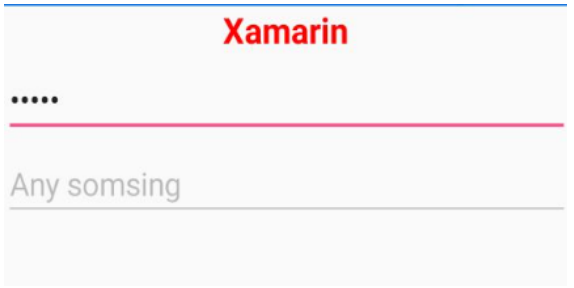

```
protected override void OnAppearing()
{
    StackLayout layout = new StackLayout();

    Label label1 = new Label();
    label1.Text = "Xamarin";
    label1.TextColor = Color.Red;
    label1.FontSize = 20;
    label1.FontAttributes = FontAttributes.Bold;
    label1.HorizontalOptions = LayoutOptions.Center;
    layout.Children.Add(label1);

    Content = layout;
}
```

Створення інтерфейсу з коду

*Для вводу тексту використовуємо елемент класу **Entry**. Застосовуємо властивість **Placeholder** (підказка для введення) та **IsPassword** = true (відображає зірочки). Також створимо ще 1 об'єкт класу Entry з іншим способом встановлення властивості.*

```
protected override void OnAppearing()
{
    StackLayout layout = new StackLayout();

    Label label1 = new Label();
    label1.Text = "Xamarin";
    label1.TextColor = Color.Red;
    label1.FontSize = 20;
    label1.FontAttributes = FontAttributes.Bold;
    label1.HorizontalOptions = LayoutOptions.Center;

    Entry input = new Entry();
    input.Placeholder = "Password";
    input.IsPassword = true;

    Entry input1 = new Entry()//інший спосіб
    {
        IsEnabled = false,
        Text = "Any somsing"
    };
    layout.Children.Add(label1);
    layout.Children.Add(input);
    layout.Children.Add(input1);

    Content = layout;
}
```

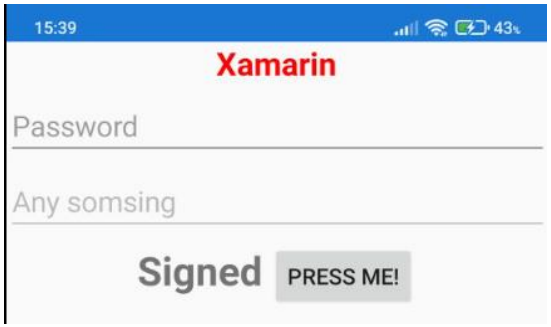




Створення інтерфейсу з коду

Створимо ще 1 StackLayout та зробимо його горизонтальним

```
protected override void OnAppearing()
        {
            StackLayout layout = new StackLayout();

            Label label1 = new Label();
            label1.Text = "Xamarin";
            label1.TextColor = Color.Red;
            label1.FontSize = 20;
            label1.FontAttributes = FontAttributes.Bold;
            label1.HorizontalOptions = LayoutOptions.Center;

            Entry input = new Entry();
            input.Placeholder = "Password";
            input.IsPassword = true;

            Entry input1 = new Entry()//інший спосіб
            {
                IsEnabled = false,
                Text = "Any somsing"
            };

            StackLayout horizontalStack = new StackLayout();
            horizontalStack.Orientation = StackOrientation.Horizontal;
            horizontalStack.HorizontalOptions = LayoutOptions.Center;

            Label sub = new Label()
            {
                Text = "Signed",
                FontSize = 25,
                FontAttributes = FontAttributes.Bold
            };

            Button button = new Button()
            {
                Text = "Press Me!"
            };

            horizontalStack.Children.Add(sub);
            horizontalStack.Children.Add(button);

            layout.Children.Add(label1);
            layout.Children.Add(input);
            layout.Children.Add(input1);
            layout.Children.Add(horizontalStack);

            Content = layout;
        }
```



Створення інтерфейсу у XAML

Створемо проект First2 на базі шаблону Blank. Переходимо до файлу MainPage.xaml. Видаляємо усе крім ContentPage. Додаємо StackLayout з відступами, створюємо Label та налаштовуємо її властивості.

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="First2.MainPage">
    <StackLayout Padding="5,5,5,5">
        <Label Text="XAMARIN" TextColor="Red"
               FontSize="25" FontAttributes="Bold"></Label>
    </StackLayout>
</ContentPage>
```

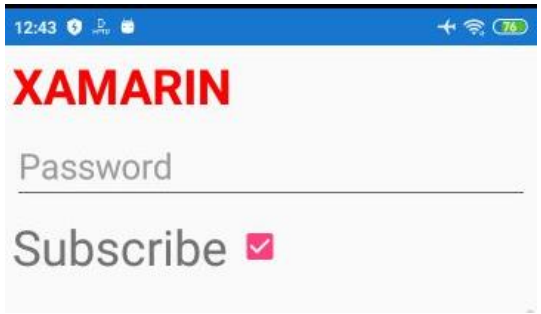
Додаємо ще елемент вводу Entry та ще один вкладений StackLayout з Label та ChexBox. Задаємо ім'я ChexBox x:Name="checkBox1", щоб до нього можна було звертатись з іншої частини програми.

```
<StackLayout Padding="5,5,5,5">
    <Label Text="XAMARIN" TextColor="Red"
           FontSize="25" FontAttributes="Bold"></Label>
    <Entry IsPassword="True" Placeholder="Password"></Entry>

    <StackLayout Orientation="Horizontal">
        <Label Text="Subscribe" FontSize="25"></Label>
        <CheckBox x:Name="checkBox1"></CheckBox>
    </StackLayout>
</StackLayout>
```

Створення інтерфейсу у XAML

Переходимо до ф-лу MainPage.xaml.cs та створюємо метод OnAppearing(), у якому встановлюємо стан checkBox1.

```
protected override void OnAppearing()
{
    // base.OnAppearing();
    checkBox1.IsChecked = true;
}
```



Самостійна робота

Створити проект калькулятор. Зміст файлу MainPage.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"
       xmlns:d="http://xamarin.com/schemas/2014/forms/design"
       xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
       mc:Ignorable="d"
       x:Class="XamarinApp1.MainPage">

        <StackLayout >
                <Label Text="Calc" Margin="5" FontSize="Large"
                       FontAttributes="Bold"/>
                <StackLayout Orientation="Horizontal" Spacing="50"
                                          HorizontalOptions="CenterAndExpand">
                        <Entry x:Name="num1" WidthRequest="100"
                               Keyboard="Numeric"/>
                        <Label Text="+"/>
                        <Entry x:Name="num2" WidthRequest="100"
                               Keyboard="Numeric"/>
                </StackLayout>
                <Label Text="=" Margin="5" HorizontalTextAlignment="Center"/>
                <Label x:Name="result" Margin="5"
                         HorizontalTextAlignment="Center"/>
                <Button Text="Calculate" Clicked="Button_Clicked"/>
        </StackLayout>
</ContentPage>
```

Самостійна робота

Зміст файлу MainPage.xaml.cs:

```
using System;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;

namespace XamarinApp1
{
    public partial class MainPage : ContentPage
    {
            public MainPage() {
                InitializeComponent();
            }

            private void Button_Clicked(object sender, EventArgs e) {
                try {
                      int n1 = Convert.ToInt32(num1.Text);
                      int n2 = Convert.ToInt32(num2.Text);
                      result.Text = (n1 + n2).ToString();
                 }
                 catch (Exception exc) { }
           }
    }
}
```