

Dimitar Minchev

Developing Cross Platform Apps

Published
with GitBook



Table of Contents

Въведение	1.1
Универсални Windows приложения	1.2
История	1.2.1
Мобилни устройства	1.2.1.1
Универсална платформа	1.2.1.2
Инсталация	1.2.2
Включване на режима за разработчици	1.2.2.1
Регистриране като разработчик на приложения	1.2.2.2
Как да проверим коя е версията на операционната система?	1.2.2.3
Приложения	1.2.3
Suma 1.0	1.2.3.1
Fibonacci 1.0	1.2.3.2
Primes 1.0	1.2.3.3
Phone Book 1.0	1.2.3.4
Clicker Mania 1.0	1.2.3.5
Clicker Mania 2.0	1.2.3.6
HTML Downloader 1.0	1.2.3.7
RSS Reader 1.0	1.2.3.8
JSON Reader 1.0	1.2.3.9
Мултиплатформени мобилни приложения	1.3
История	1.3.1
Приложения	1.3.2
Suma 2.0	1.3.2.1
Fibonacci 2.0	1.3.2.2
Primes 2.0	1.3.2.3
Phone Book 2.0	1.3.2.4
Clicker Mania 3.0	1.3.2.5
HTML Downloader 2.0	1.3.2.6
RSS Reader 2.0	1.3.2.7
JSON Reader 2.0	1.3.2.8
Заключение	1.4
Бележка	1.4.1
Книга	1.4.2
Автор	1.4.3
Лиценз	1.4.4
Формат	1.4.5

Developing Cross-Platform Apps

Настоящият проект **Developing Cross-Platform Apps** представя Microsoft технологиите за разработка на **Универсални Windows приложения и Мултиплатформени мобилни приложения**. Учебното пособие е налично за свободно четене под формата на [електронна книга](#). Учебните ресурси са налични за свободно изтегляне от [електронно хранилище](#). Всички материали се разпространяват под лиценз [CC-BY-NC-SA](#).

Формат	ISBN
PDF	978-619-7126-66-2
MOBI	978-619-7126-67-9
EPUB	978-619-7126-68-6

Автор

Димитър Минчев е университетски преподавател към Център по информатика и технически науки при Бургаски свободен университет. Подготвя студенти за [Републиканската студентска олимпиада по програмиране](#) в [Клуб по състезателно програмиране](#). Организира съревнование за разработка на настолни и мобилни приложения [ХАКАТОН @ БСУ](#). Създава уникалните [Академията за таланти по програмиране](#) и [Школа по роботика](#) за ученици от Бургас. Инициира ученическото състезание по програмиране [CODE@BURGAS](#). Участва в националната програма [Обучение за ИТ умения и кариера](#) на Министерството на образованието и науката.

Контакт
Служебен: тел. +359 56 900 477 и e-mail: mitko@bfu.bg
Личен: моб. +359 899 148 872 и e-mail: dimitar.minchev@gmail.com
Блог: http://www.minchev.eu

Универсални Windows приложения

Информация

Настоящата глава от книгата въвежда читателя във вълнуващия свят на Microsoft технологиите за разработка на универсални приложения за платформа Windows.

Съвет

Пожелавам Ви много забавление, провали и успехи използвайки Microsoft технологиите за разработка на универсални Windows приложения.

Забележка

Примерите в тази част са разработени използвайки следните версии на операционната система и интегрираната среда за разработка:

- Microsoft Windows 11 Version 10.0.22000.282
- Microsoft Visual Studio Community 2022 Version 17.0.0

История

Компанията Microsoft е транснационална компания, развиваща дейност в областта на компютърните технологии и разработката на софтуер.

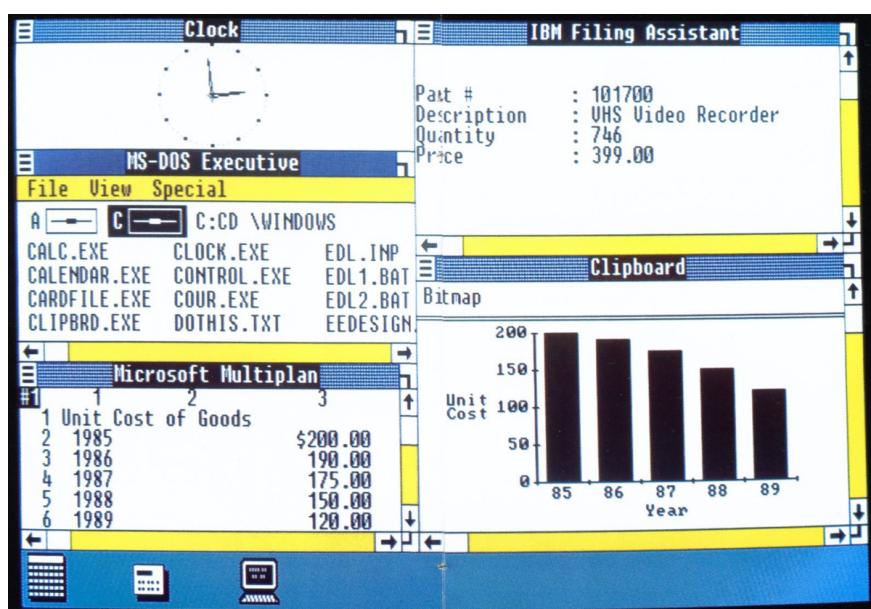
Седалището ѝ е разположено в гр. Редмънд, Съединени американски щати.

Основана е от Бил Гейтс и Пол Альн през 1975 година.



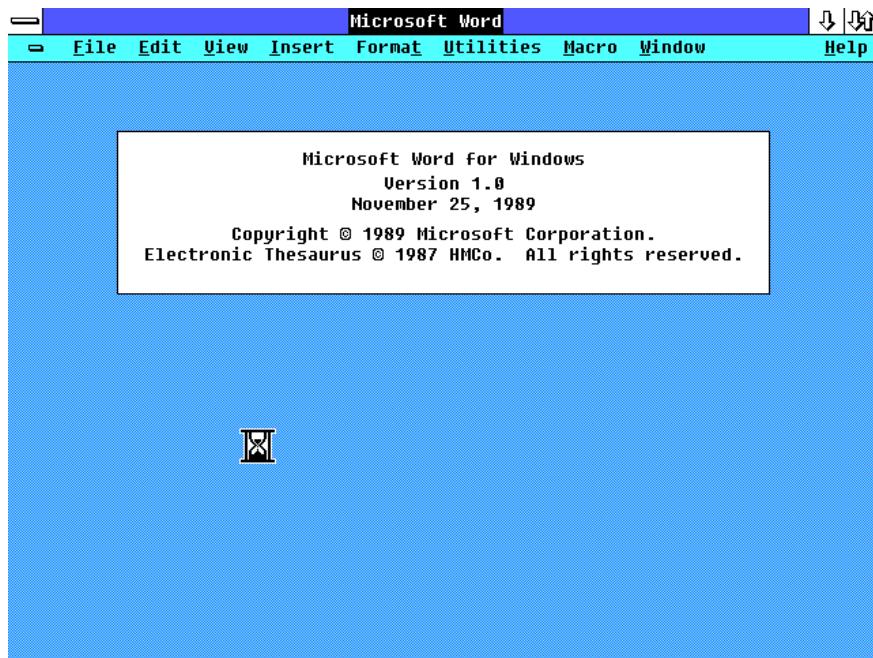
Фиг. 1. Пол Альн (отляво) и Бил Гейтс (отдясно) основават Microsoft през 1975

Първата версия на операционна система MS-DOS излиза през Август 1981, а графичния интерфейс Windows 1.0 на 20 ноември 1985. Първата версия на популярния пакет от приложения за офиса Microsoft Office излиза на 19 Ноември 1990 г.



Фиг. 2. Microsoft Windows 1.0 излиза през 1983

Мобилни устройства



Фиг. 3. Microsoft Word 1.0 излиза през 1989

Операционните системи Windows 8 и 8.1 са представени съответно на 1 Август 2012 и 27 Август 2013. Поддръжката за Windows 8 и 8.1 се преустановява съответно на 12 Януари 2016 и 10 Януари 2023. Последната версия на най-използваната компютърна операционната система Windows 10 излиза на 15 Юли 2015.



Фиг. 4. Windows 8.1 излиза през 2013

Мобилни устройства



Фиг. 5. Windows 10 излиза през 2015

По официални данни обявени от компанията Microsoft, инсталациите на Windows 10 за периода 2015 - 2020 са показани в Табл. 1.

Дата	Устройства (Милиони)
Юли 2015	14
Август 2015	75
Октомври 2015	110
Януари 2016	200
Март 2015	270
Май 2016	300
Юли 2016	350
Септември 2016	400
Май 2017	500
Ноември 2017	600
Май 2018	700
Март 2019	800
Септември 2019	900
Март 2020	1000
2021	1300

Табл. 1 Инсталации на операционната система Windows 10

По официални данни обявени от компанията Microsoft, публичните версии на операционната система Windows 10 са показани в Табл. 2

Version	Build	Release
1507	10240	July 2015
1511	10586	November 2015
1607	14393	August 2016
1703	15063	April 2017
1709	16299	October 2017
1803	17134	April 2018
1809	17763	October 2018
1903	18362	May 2019
1909	18363	November 2019
2004	19041	May 2020
20H2	19042	October 2020
21H1	19043	May 2021

Табл. 2 Публични версии на операционната система Windows 10

Повече информация за версията на операционната система Windows 10, може да бъде намерена в Интернет на адрес: <https://docs.microsoft.com/en-us/windows/release-health/release-information/>

Мобилни устройства

Windows Phone е операционна система за мобилни устройства (смартфони), разработена от Microsoft, наследяваща платформата Windows Mobile.

Windows Phone предлага една изцяло нова визия и функционалност, в крак с модерния начин на живот и постоянно свързаност към Интернет, която е добре комбинирана с мощн хардуер, осигуряващ оптимален комфорт при ползване с разнообразна мултимедия.



Фиг.6. Windows Phone 7 излиза през 2010

Първият Windows Phone 7 е представен на 21 октомври 2010, локализиран е на 25 езика, достъпен в 35 страни, а приложенията за него се публикуват в Windows Phone Store. Базиран е на Windows Embedded Compact 7 версията на Windows Embedded CE. Поддръжката се преустановява на 14 октомври 2014.



Фиг. 7. Windows Phone 8 излиза през 2012

Втората генерация мобилно устройство Windows Phone 8 излиза на 29 октомври 2012, а Windows Phone 8.1 на 2 април 2014. Windows CE се заменя от Windows NT кернела от Windows 8. Приложенията за него се публикуват в

Мобилни устройства

Windows Phone Store. Поддръжката на Windows Phone 8 се преустановява на 12 Януари 2016, а на Windows Phone 8.1 на 11 Юли 2017 .

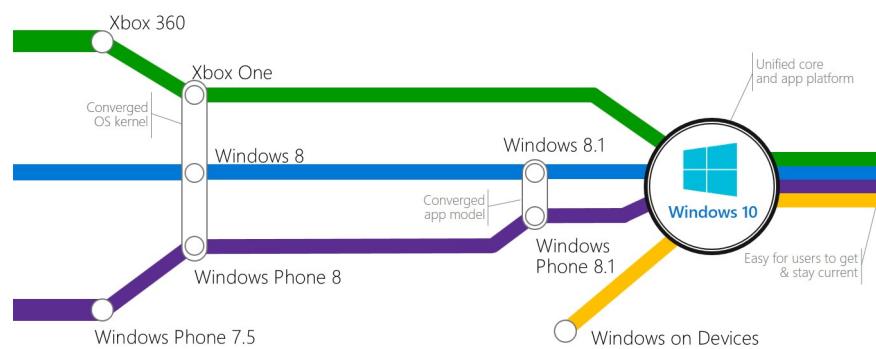


Фиг. 8. Windows 10 Mobile излиза през 2015

На 20 ноември 2015 е представена операционната система Windows 10 Mobile. Поддържа се архитектурата Universal Windows Platform. Приложенията се публикуват и изтеглят от Windows Store. Поддръжката се преустановява на 11 Юни 2019.

Универсална платформа

На 13 септември 2011 по време на конференцията Build, бе представен магазина за приложения Windows Store, предназначен за разпространение на приложения за компютри с инсталирана операционната система Windows 8.



Фиг. 9. История на развитието на магазина за приложения на Microsoft

При представянето на платформата Windows Phone през 2012, Windows Phone Marketplace, бе преименуван на Windows Phone Store, като този магазин осигурява разпространението на приложения за мобилни устройства Windows Phone.

С излизането на Windows 10, магазините за приложения Windows Phone Store и Windows Store, баха обединени. Операционната система премина към единна архитектура наречена Windows Universal Platform. Днес магазина за приложения на компанията се нарича Microsoft Store.



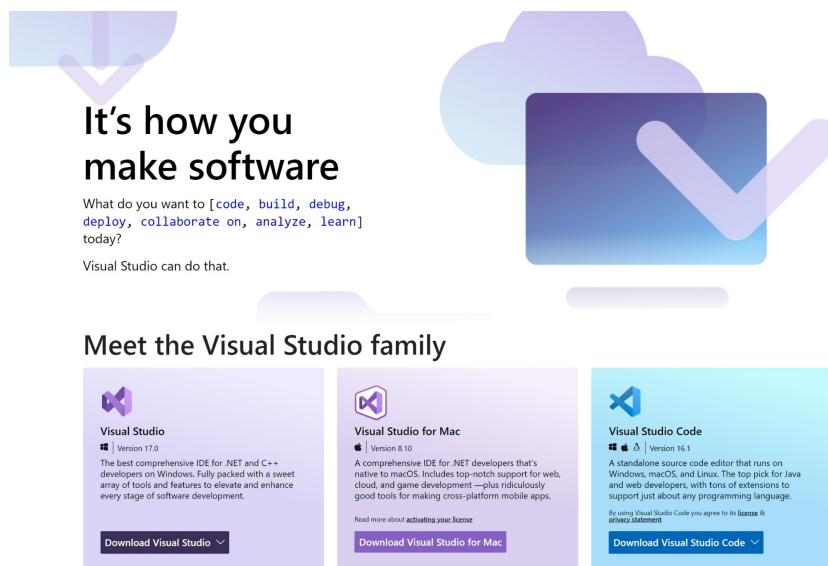
Фиг.10. Архитектура на универсалната Windows платформа за приложения

Настоящото пособие въвежда читателя във вълнуващия свят на Microsoft технологии за разработка на универсални приложения за платформа Windows.

Инсталация

Напълно бесплатна версия на интегрираната среда за разработка наречена **Microsoft Visual Studio Community Edition**, може да се изтегли от официалният сайт на компанията в Интернет на адрес:

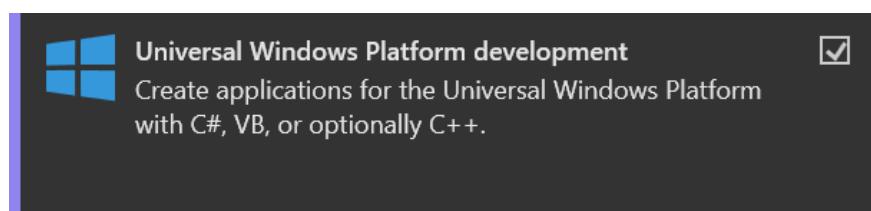
<https://visualstudio.microsoft.com/>. Понастоящем последната актуална версия на този продукт е Microsoft Visual Studio Community Edition 2022.



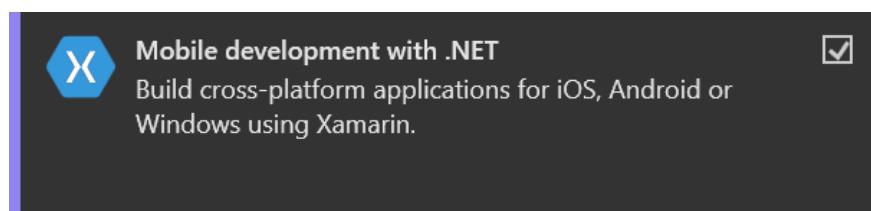
Фиг. 11. Microsoft Visual Studio Website

Стандартна инсталация по подразбиране изиска скромните **1.39 GB** свободно дисково пространство, но тя включва само редактора на интегрираната среда за разработка Visual Studio, както и инструменти за колаборативна работа.

За разработка на универсални Windows приложения е необходимо да се включи опцията **Universal Windows Platform development** (Фиг. 12) и за разработка на мултиплатформени-мобилни приложения опцията **Mobile development with .NET** (Фиг. 13).



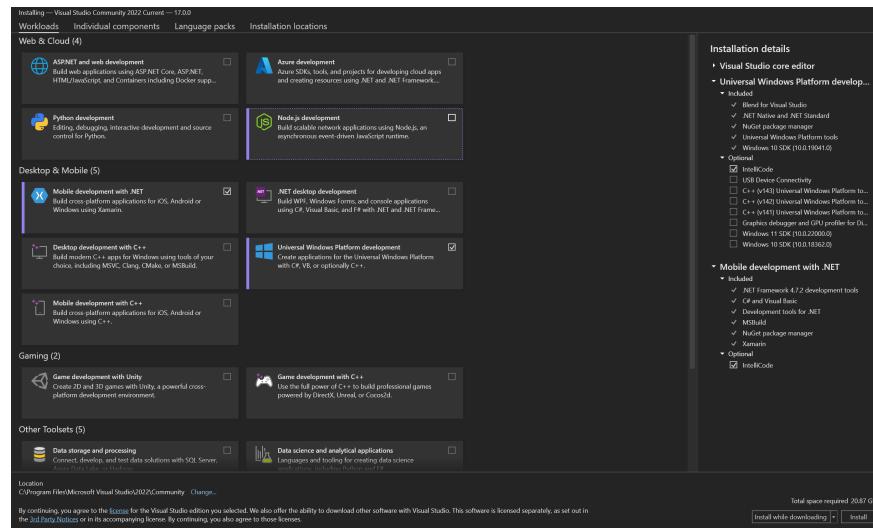
Фиг. 12. Visual Studio Installer: Universal Windows Platform development



Мобилни устройства

Фиг. 13. Visual Studio Installer: Mobile development with .NET

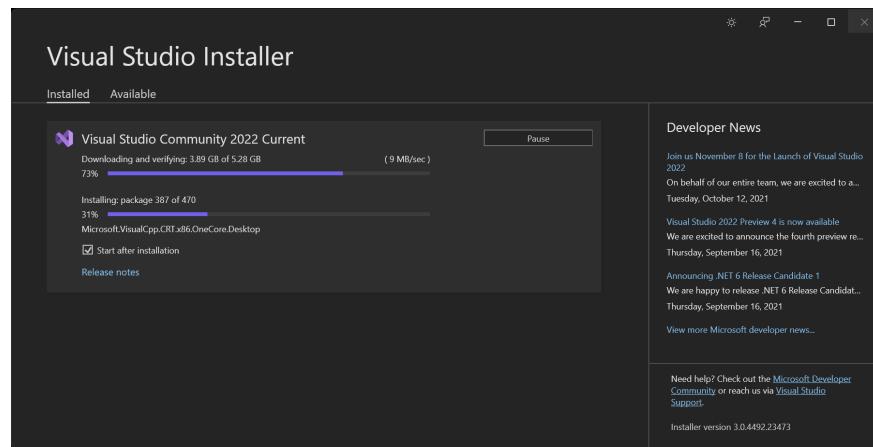
Допълнителните инструменти за разработчици увеличават обема на необходимото дисково пространство за инсталiranе на интегрираната среда за разработка Visual Studio. Необходими са **20.87 GB** свободно дисково пространство (Фиг. 14).



Фиг. 14. Visual Studio Setup

Продължава се напред и се изчаква приключването на инсталацията.

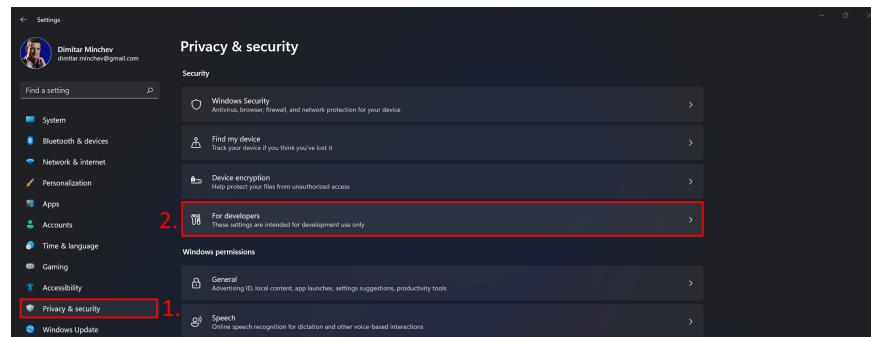
Процедурата отнема значително количество време.



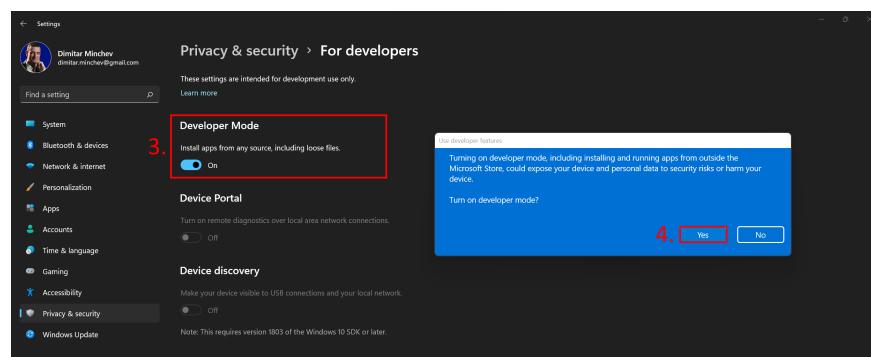
Фиг. 15. Visual Studio Installer

Включване на режима за разработчици

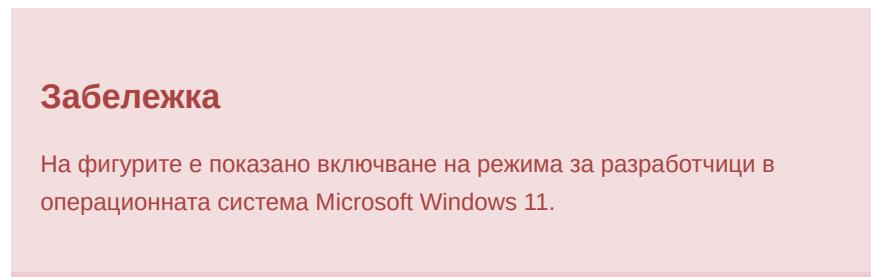
За да разработвате универсални приложения за платформа Windows e необходимо да включите операционната система в режим за разработчици, както е показано на Фиг. 16 (a) и Фиг. 16 (b).



Фиг.16 (a). Включване на режима за разработчици



Фиг.16 (b) Включване на режима за разработчици

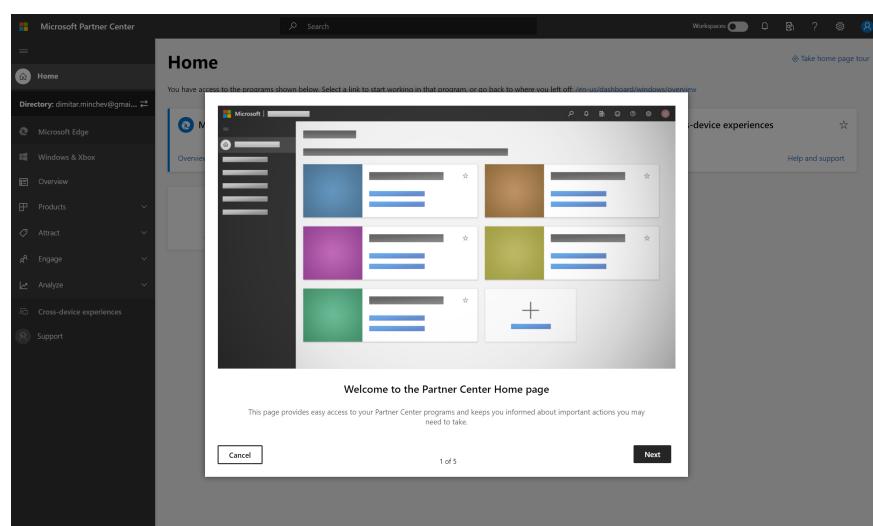


Регистриране като разработчик на приложения

За да публикувате приложения за Windows платформата е необходимо да се регистрирате като разработчик в центъра за разработчици, наречен **Microsoft Partner Center**. Порталът предоставя възможността за публикуване в магазина за приложения наречен **Microsoft Store**.

Регистрацията е достъпна в Интернет на адрес: <https://developer.microsoft.com/en-us/microsoft-store/register/>.

Регистрацията като разработчик на приложения в центъра за разработчици се заплаща с еднократна такса от **19** американски долара за индивидуални потребители и **99** американски долара за корпоративни клиенти.



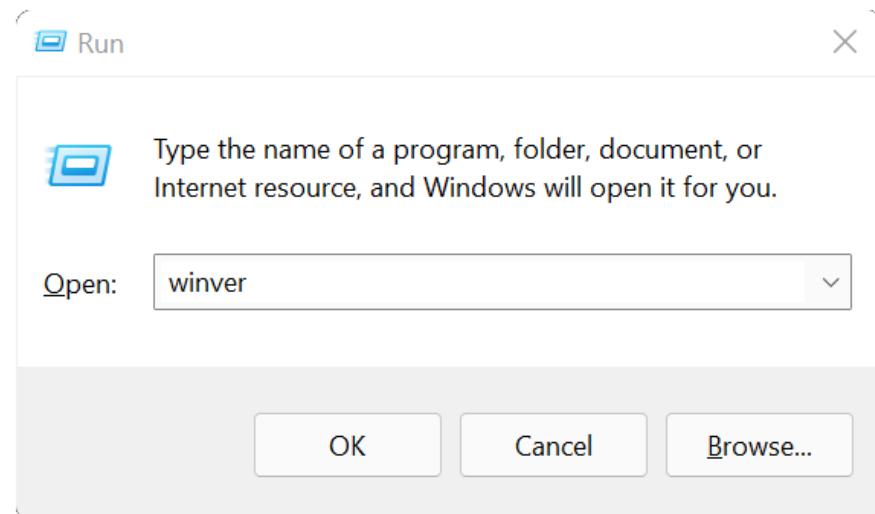
Фиг. 17. Център за разработчици на Microsoft

Как да проверим коя е версията на операционната система?

За да проверите текущата версия на операционната система Windows използвайте клавишина комбинация Windows Key + R (Фиг. 18). В появилия се диалогов прозорец RUN запишете команда WINVER (Фиг. 19) и натиснете Enter. В резултат ще видите екран About Windows (Фиг. 20) в който се вижда текущата версията на операционната система.



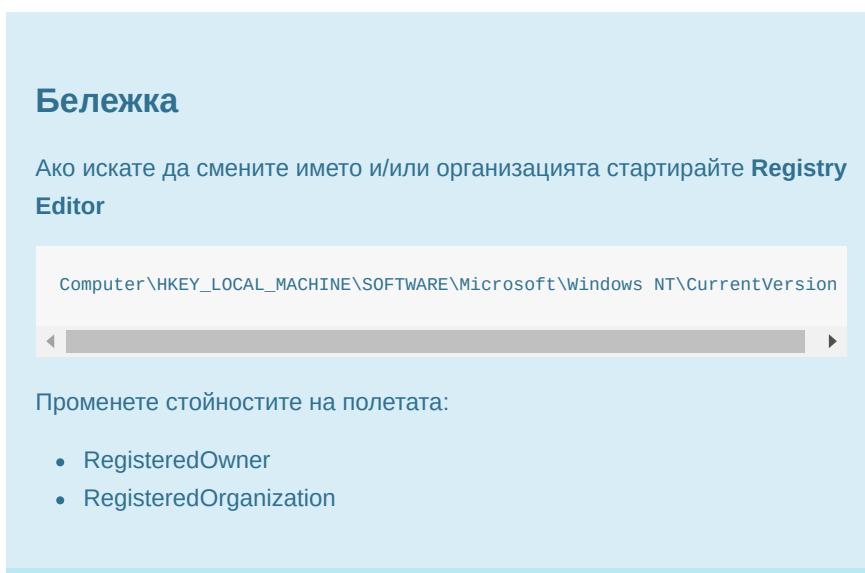
Фиг.18. Клавишина комбинация Windows Key + R



Фиг.19. Диалогов прозорец RUN



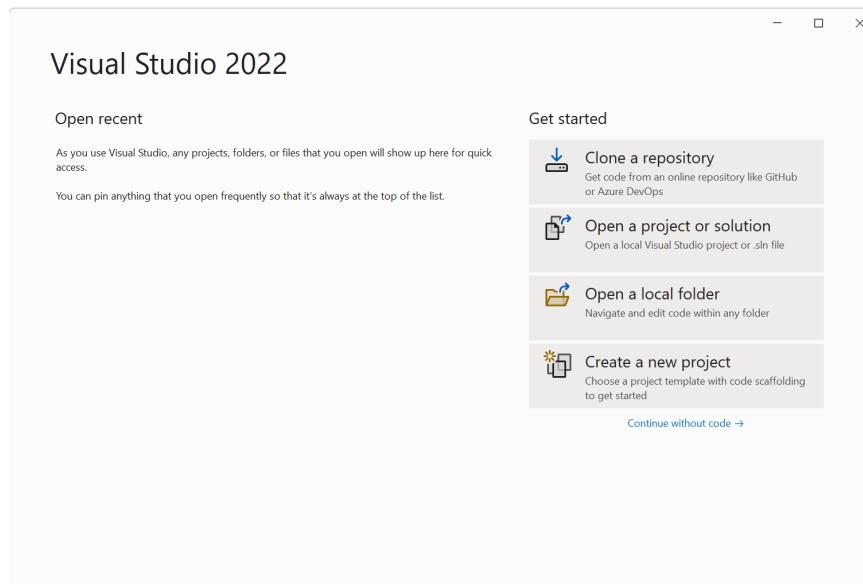
Фиг.20. Версия на операционната система Windows



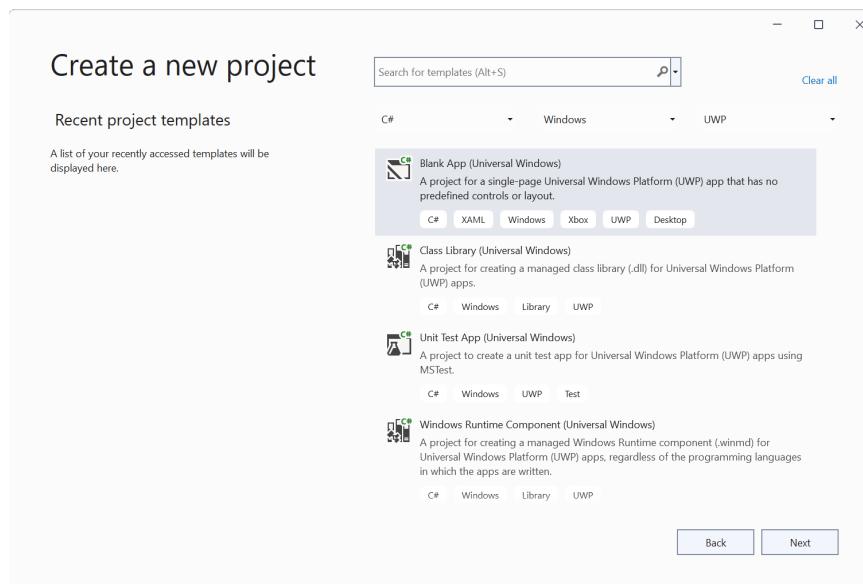
Приложения

В тази част са представени редица приложения демонстриращи разработка на универсални приложения за платформа Windows.

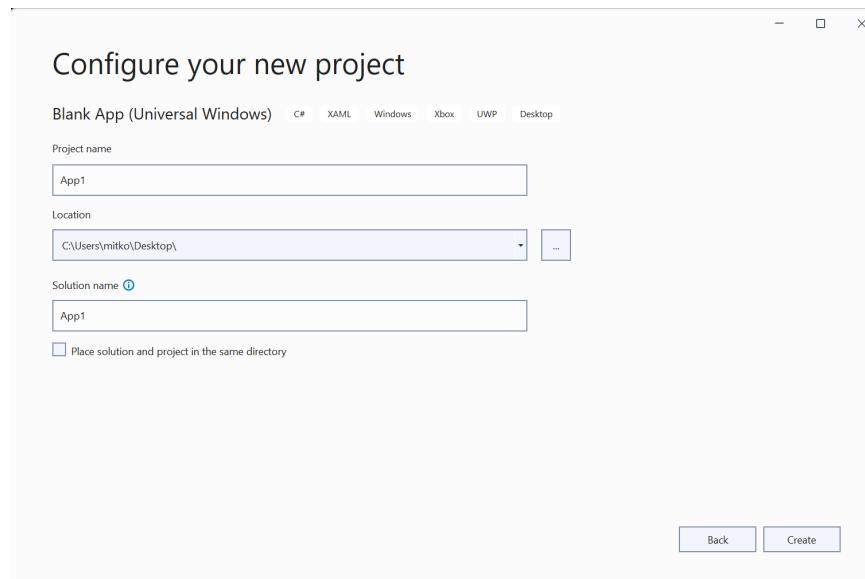
Разработката на всяко приложение в интегрираната среда за разработка Microsoft Visual Studio, започва със създаване на проект. Нов проект се създава от менюто посредством изпълняване на последователността: **File > New > Project**. Може да се използва и съкратената клавишка комбинация **Ctrl + Shift + N** за създаване на нов проект.



Фиг. 21 (a). Създаване на нов проект в интегрираната среда за разработка Visual Studio



Фиг. 21 (b). Създаване на нов проект в интегрираната среда за разработка Visual Studio



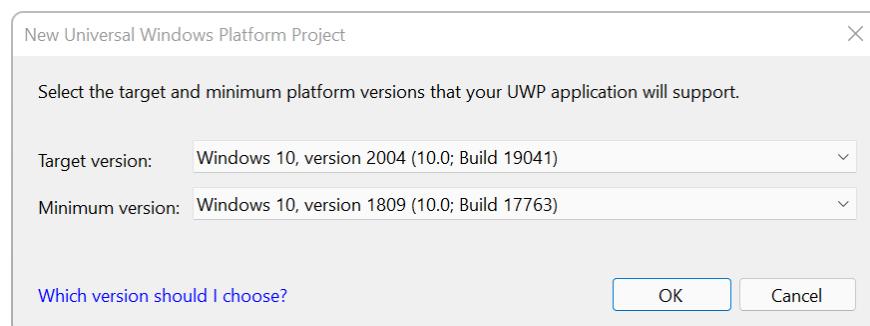
Фиг. 21 (с). Създаване на нов проект в интегрираната среда за разработка Visual Studio

Suma 1.0

Използвайки интегрираната среда за разработка Visual Studio и езика за програмиране C# ще разработим универсално приложение за намиране сума на две числа.

Start

Стартирайте интегрираната среда за разработка Visual Studio. Създайте нов проект от менюто посредством изпълняване на последователността: **File > New > Project** или използвайте съкратената клавишка комбинация **Ctrl + Shift + N**. В появлия се диалогов прозорец изберете: **Visual C# > Windows Universal > Blank App (Universal Windows)**. За име на проекта запишете: **Suma 1.0**.



Фиг. 22. Избор на Windows версия

От Solution Explorer отворете файловете **MainPage.xaml** и **MainPage.xaml.cs**. В случай, че не виждате Solution Explorer можете да го отворите от менюто **View > Solution Explorer** или като използвате съкратената клавишка последователност **Ctrl + W, S**.

MainPage.xaml

Файлът **MainPage.xaml** съдържа изходния код от дизайна на потребителския интерфейс на разработваното приложение и се пише на езика XAML. Копирайте (**Ctrl+C**) и поставете (**Ctrl+V**) програмният фрагмент даден по-долу във Вашето приложение.

```
<Page
    x:Class="Suma_1._0.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:Suma_1._0"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <!-- User Interface (UI): Suma 1.0 -->
    <StackPanel Padding="50" Background="Orange">

        <!-- Title -->
        <TextBlock Text="Suma 1.0" FontSize="42" />

        <!-- A -->
        <TextBlock Text="A=" FontSize="24" />
        <TextBox x:Name="boxA" FontSize="24" />

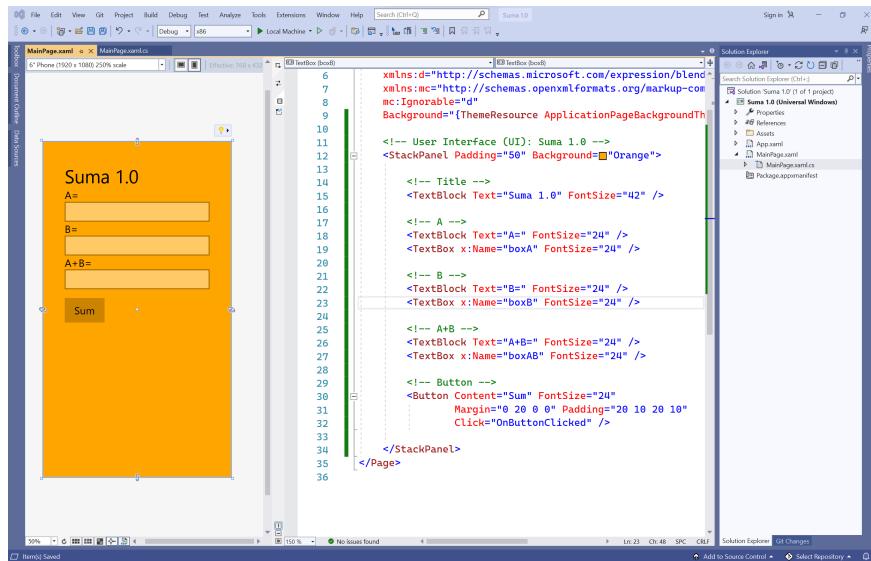
        <!-- B -->
        <TextBlock Text="B=" FontSize="24" />
        <TextBox x:Name="boxB" FontSize="24" />

        <!-- A+B -->
        <TextBlock Text="A+B=" FontSize="24" />
        <TextBox x:Name="boxAB" FontSize="24" />

        <!-- Button -->
        <Button Content="Sum" FontSize="24"
            Margin="0 20 0 0" Padding="20 10 20 10"
            Click="OnButtonClicked" />

    </StackPanel>
</Page>
```

Изглед от дизайна на потребителският интерфейс (**XAML**) в интегрираната среда за разработка Visual Studio по време на разработване на приложението:



Фиг. 23. Изглед от дизайна на потребителският интерфейс

MainPage.xaml.cs

Файлът **MainPage.xaml.cs** съдържа изходния код от бизнес логиката на разработваното приложение и се пише на програмният език C#. Копирайте (**Ctrl+C**) и поставете (**Ctrl+V**) програмният фрагмент даден по-долу във Вашето приложение.

```

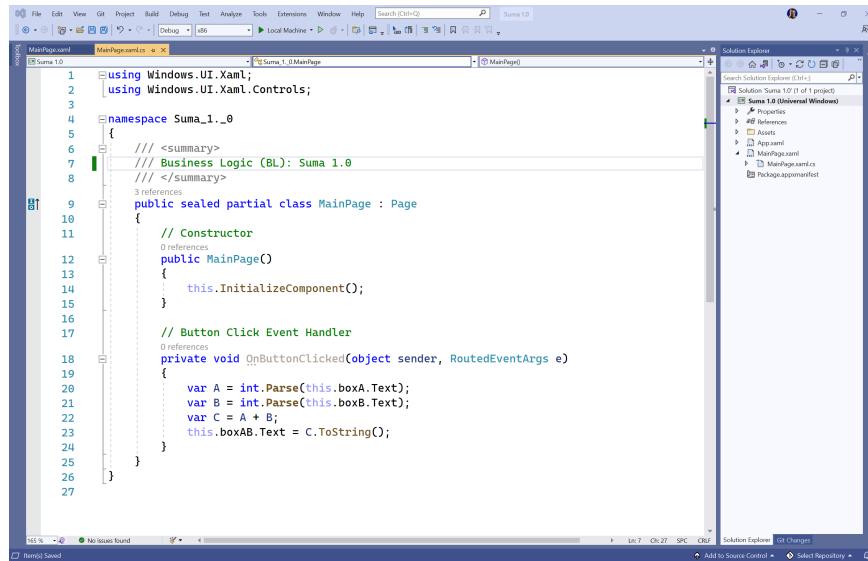
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;

namespace Suma_1._0
{
    /// <summary>
    /// Business Logic (BL): Suma 1.0
    /// </summary>
    public sealed partial class MainPage : Page
    {
        // Constructor
        public MainPage()
        {
            this.InitializeComponent();
        }

        // Button Click Event Handler
        private void OnButtonClicked(object sender, RoutedEventArgs e)
        {
            var A = int.Parse(this.boxA.Text);
            var B = int.Parse(this.boxB.Text);
            var C = A + B;
            this.boxAB.Text = C.ToString();
        }
    }
}

```

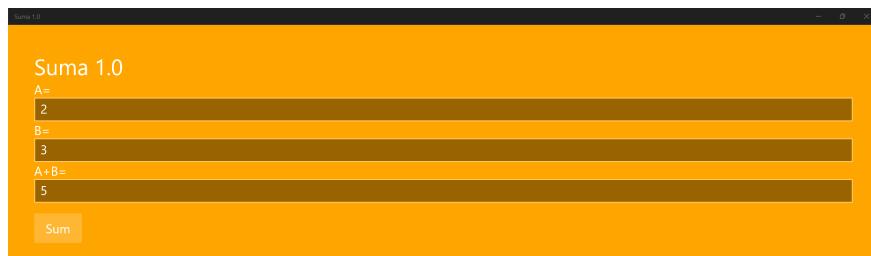
Изглед от бизнес логиката (C#) в интегрираната среда за разработка Visual Studio по време на разработване на приложението:



Фиг. 24. Изглед от бизнес логиката на разработваното приложение

Demo

Стартирайте приложението от менюто: **Debug > Start Debugging** или като натиснете клавиш **F5**.



Фиг. 25. Универсално приложение за намиране сума на две числа

Fibonacci 1.0

Използвайки интегрираната среда за разработка Visual Studio и езика за програмиране C# ще разработим универсално приложение загенериране на числата от редицата на Фибоначи.

Информация

Числата на Фибоначи в математиката образуват редица, която се дефинира рекурсивно по следния начин: започва се с 0 и 1, а всеки следващ член на редицата се получава като сума на предходните два. Първите числа на Фибоначи са: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

- Източник: [Wikipedia](#)

Start

Стартирайте интегрираната среда за разработка **Visual Studio**. Създайте нов проект от менюто посредством изпълняване на последователността: **File > New > Project** или използвайте съкратената клавишна комбинация **Ctrl + Shift + N**. В появилния се диалогов прозорец изберете: **Visual C# > Windows Universal > Blank App (Universal Windows)**. За име на проекта запишете: **Fibonacci 1.0**. От Solution Explorer отворете файловете **MainPage.xaml** и **MainPage.xaml.cs**. В случай, че не виждате Solution Explorer можете да го отворите от менюто **View > Solution Explorer** или като използвате съкратената клавишна последователност **Ctrl + W, S**.

MainPage.xaml

Файлът **MainPage.xaml** съдържа изходния код от дизайна на потребителския интерфейс на разработваното приложение и се пише на езика XAML. Копирайте (**Ctrl+C**) и поставете (**Ctrl+V**) програмният фрагмент даден по-долу във Вашето приложение.

```

<Page
    x:Class="Fibonacci_1_0.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:Fibonacci_1_0"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <!-- User Interface (UI): Fibonacci 1.0 -->
    <StackPanel Background="Pink" Padding="50">

        <!-- Title -->
        <TextBlock Text="Fibonacci 1.0" FontSize="40" />

        <!-- Limit -->
        <TextBlock Text="Limit" Margin="0 10 0 10" FontSize="20" />
        <TextBox Name="boxLimit" FontSize="20" Text="1000" />
        <Button Content="Generate" Margin="0 10 0 10" Padding="20 10 20 10" For

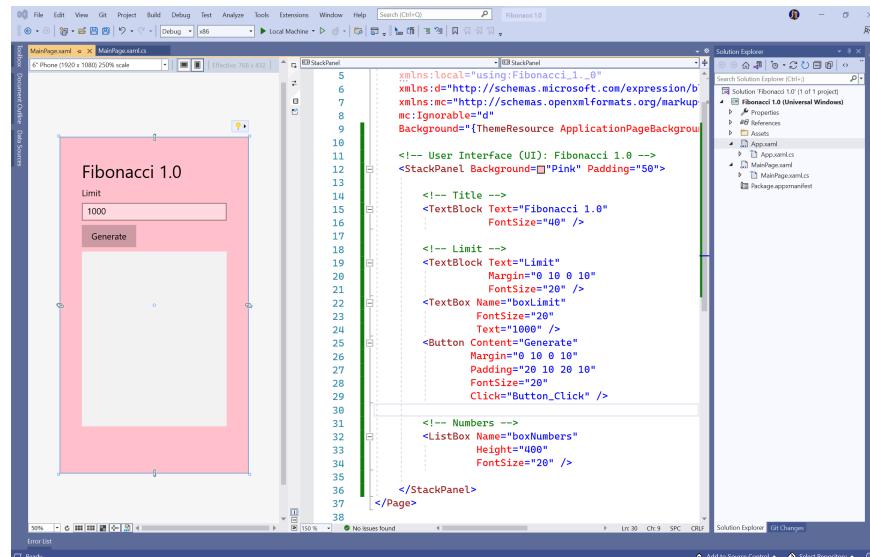
        <!-- Numbers -->
        <ListBox Name="boxNumbers" Height="400" FontSize="20" />

    </StackPanel>
</Page>

```



Изглед от дизайна на потребителският интерфейс (XAML) в интегрираната среда за разработка Visual Studio по време на разработване на приложението:



Фиг. 26. Изглед от дизайна на потребителският интерфейс

MainPage.xaml.cs

Файлът **MainPage.xaml.cs** съдържа изходния код от бизнес логиката на разработваното приложение и се пише на програмният език C#. Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

```

using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using System.Collections.ObjectModel;

namespace Fibonacci_1._0
{
    /// <summary>
    /// Business Logic (BL): Fibonacci 1.0
    /// </summary>
    public sealed partial class MainPage : Page
    {
        // Collection
        private ObservableCollection<int> numbers = new ObservableCollection<int>();

        // Constructor
        public MainPage()
        {
            this.InitializeComponent();
            boxNumbers.ItemsSource = numbers;
        }

        // Button Click Event Handler
        private void Button_Click(object sender, RoutedEventArgs e)
        {
            numbers.Add(1);
            numbers.Add(1);
            int limit = int.Parse(boxLimit.Text);
            int a = 1, b = 1, c = a + b;
            while (c < limit)
            {
                numbers.Add(c);
                a = b;
                b = c;
                c = a + b;
            }
        }
    }
}

```



Изглед от бизнес логиката (C#) в интегрираната среда за разработка Visual Studio по време на разработване на приложението:

The screenshot shows the Visual Studio interface with the code editor open. The file is MainPage.xaml.cs, containing C# code for a Universal Windows Platform (UWP) application. The code implements a Fibonacci sequence generator. The Solution Explorer on the right shows the project structure for 'Fibonacci 1.0'.

```

1 //<summary>
2 /// Business Logic (BL): Fibonacci 1.0
3 //</summary>
4
5 public sealed partial class MainPage : Page
6 {
7     // Collection
8     private ObservableCollection<int> numbers = new ObservableCollection<int>();
9
10    // Constructor
11    public MainPage()
12    {
13        this.InitializeComponent();
14        boxNumbers.ItemsSource = numbers;
15    }
16
17    // Button Click Event Handler
18    private void button_Click(object sender, RoutedEventArgs e)
19    {
20        numbers.Add(1);
21        numbers.Add(1);
22        int limit = int.Parse(boxLimit.Text);
23        int a = 1, b = 1, c = a + b;
24        while (c < limit)
25        {
26            numbers.Add(c);
27            a = b;
28            b = c;
29            c = a + b;
30        }
31    }
32}
33
34}
35
36}
37

```

Фиг. 27. Изглед от бизнес логиката на разработваното приложение

Demo

Стартирайте приложението от менюто: **Debug > Start Debugging** или като натиснете клавиш **F5**.



Фиг. 28. Универсално приложение за генериране на числата от редицата на Фибоначи.

Primes 1.0

Използвайки интегрираната среда за разработка Visual Studio и езика за програмиране C# ще разработим универсално приложение за генериране на редица от прости числа.

Информация

Просто число е естествено число, по-голямо от 1, което не е произведение на две по-малки естествени числа.

- Източник: [Wikipedia](#)

Start

1. Стаптирайте интегрираната среда за разработка **Visual Studio**.
2. Създайте нов проект **Visual C# > Windows Universal > Blank App (Universal Windows)**.
3. За име на проекта запишете: **Primes 1.0**.

MainPage.xaml

Файлът **MainPage.xaml** съдържа изходния код от дизайна на потребителския интерфейс на разработваното приложение и се пише на езика XAML.
Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

```

<Page
    x:Class="Primes_1._0.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:Primes_1._0"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <!-- User Interface (UI): Primes 1.0 -->
    <StackPanel Background="Yellow" Padding="50">

        <!-- Title -->
        <TextBlock Text="Primes 1.0" FontSize="40" />

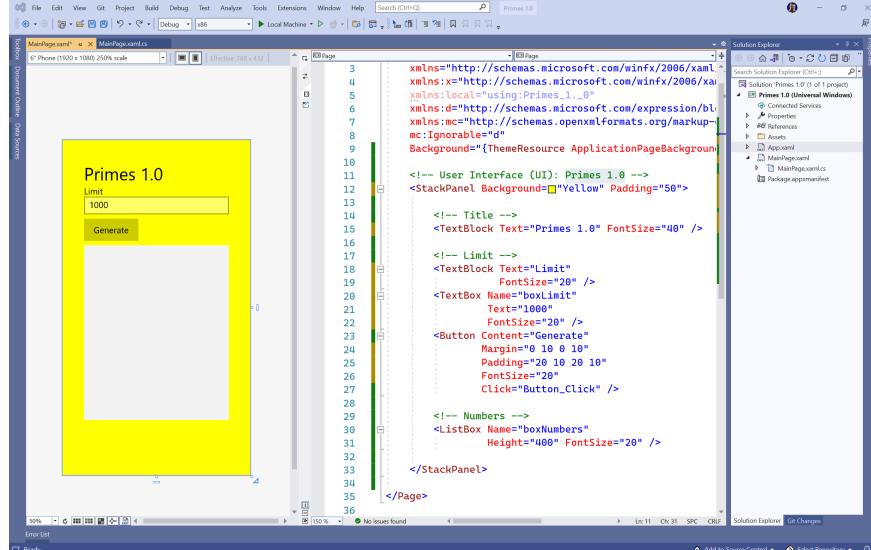
        <!-- Limit -->
        <TextBlock Text="Limit" FontSize="20" />
        <TextBox Name="boxLimit" Text="1000" FontSize="20" />
        <Button Content="Generate" Margin="0 10 0 10" Padding="20 10 20 10" For

        <!-- Numbers -->
        <ListBox Name="boxNumbers" Height="400" FontSize="20" />

    </StackPanel>
</Page>

```

Изглед от дизайна на потребителският интерфейс (XAML) в интегрираната среда за разработка Visual Studio по време на разработване на приложението:



Фиг. 29. Изглед от дизайна на потребителският интерфейс

MainPage.xaml.cs

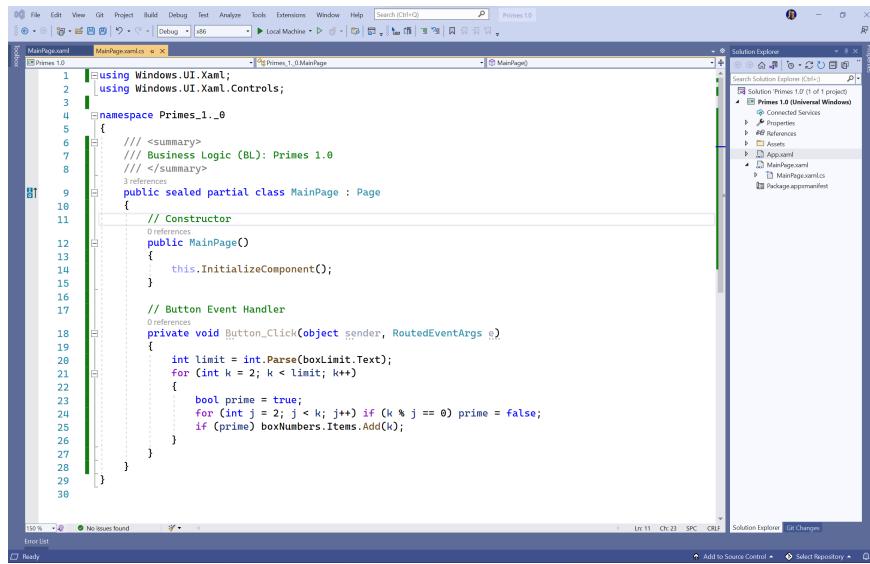
Файлът **MainPage.xaml.cs** съдържа изходния код от бизнес логиката на разработваното приложение и се пише на програмният език C#. Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

```
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;

namespace Primes_1._0
{
    /// <summary>
    /// Business Logic (BL): Primes 1.0
    /// </summary>
    public sealed partial class MainPage : Page
    {
        // Constructor
        public MainPage()
        {
            this.InitializeComponent();
        }

        // Button Event Handler
        private void Button_Click(object sender, RoutedEventArgs e)
        {
            int limit = int.Parse(boxLimit.Text);
            for (int k = 2; k < limit; k++)
            {
                bool prime = true;
                for (int j = 2; j < k; j++) if (k % j == 0) prime = false;
                if (prime) boxNumbers.Items.Add(k);
            }
        }
    }
}
```

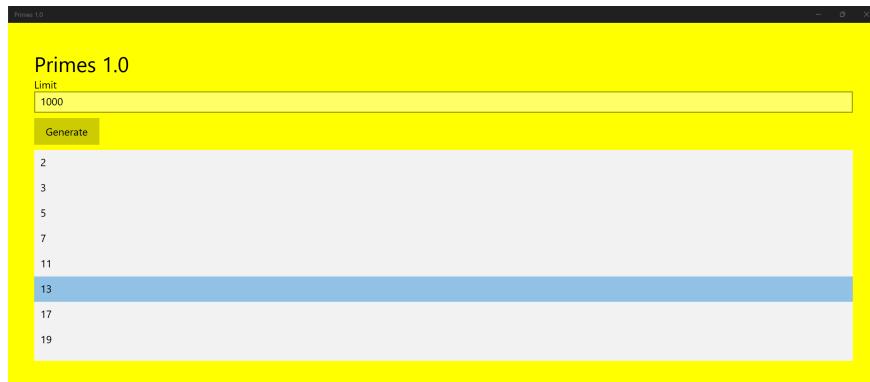
Изглед от бизнес логиката (C#) в интегрираната среда за разработка Visual Studio по време на разработване на приложението:



Фиг. 30. Изглед от бизнес логиката на разработваното приложение

Demo

Стартирайте приложението от менюто: **Debug > Start Debugging** или като натиснете клавиши **F5**.



Фиг. 31. Универсално приложение за генериране на редица от прости числа.

Phone Book 1.0

Използвайки интегрираната среда за разработка Visual Studio и езика за програмиране C# ще разработим универсално приложение тип телефонен указател съдържащо списък с контакти. Изходния код от дизайна на потребителския интерфейс (XAML) и бизнес логиката (C#) на приложението са дадени в програмните фрагменти по-долу:

Contact.cs

```
using System;
namespace Phone_Book_1._0
{
    public class Contact
    {
        public Uri picture { get; set; }
        public string name { get; set; }
        public string phone { get; set; }

        public Contact(Uri _picture, string _name, string _phone)
        {
            this.picture = _picture;
            this.name = _name;
            this.phone = _phone;
        }
    }
}
```

App.xaml.cs

```
using System;
using System.Collections.ObjectModel;
using Windows.ApplicationModel;
using Windows.ApplicationModel.Activation;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Navigation;

namespace Phone_Book_1_0
{
    /// <summary>
    /// Provides application-specific behavior to supplement the default Application class.
    /// </summary>
    sealed partial class App : Application
    {
        // Contacts
        public static ObservableCollection<Contact> contacts = new ObservableCollection<Contact>();

        ...
    }
}
```

AddPage.xaml

```
<Page
    x:Class="Phone_Book_1._0.AddPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:Phone_Book_1._0"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <!-- User Interface (UI): Phone Book 1.0 -->
    <StackPanel Background="Cyan" Padding="40">

        <!-- Title -->
        <TextBlock Text="Add Contact" FontSize="40" />

        <!-- Name -->
        <TextBlock Text="Name" FontSize="20" />
        <TextBox Name="boxName" FontSize="20" />

        <!-- Phone -->
        <TextBlock Text="Phone" FontSize="20" />
        <TextBox Name="boxPhone" FontSize="20" />

        <!-- Picture -->
        <TextBlock Text="Picture" FontSize="20" />
        <TextBox Name="boxPicture" FontSize="20" />

        <!-- Button -->
        <Button Content="Add Contact" FontSize="20" Padding="20 10 20 10" Margin="0 10 0 0" />

    </StackPanel>
</Page>
```

AddPage.xaml.cs

```
using System;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;

namespace Phone_Book_1_0
{
    /// <summary>
    /// Business Logic (BL): Phone Book 1.0
    /// </summary>
    public sealed partial class AddPage : Page
    {
        // Constructor
        public AddPage()
        {
            this.InitializeComponent();
        }

        // Button Click Event Handler
        private void Button_Click(object sender, RoutedEventArgs e)
        {
            var picture = new Uri(boxPicture.Text);
            var contact = new Contact(picture, boxName.Text, boxPhone.Text);
            this.Frame.Navigate(typeof(MainPage), contact);
        }
    }
}
```

MainPage.xaml

```
<Page
    x:Class="Phone_Book_1._0.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:Phone_Book_1._0"
    xmlns:d="http://schemas.microsoft.com/expressionblend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <!-- User Interface (UI): Phone Book 1.0 -->
    <StackPanel Background="Cyan" Padding="40">

        <!-- Title -->
        <TextBlock Text="Phone Book 1.0" FontSize="40" />

        <!-- Contacts -->
        <ListBox Name="boxContacts" Height="400">
            <ListBox.ItemTemplate>
                <DataTemplate>
                    <StackPanel Orientation="Horizontal">
                        <Image Source="{Binding picture}" Width="100" Height="100" />
                        <StackPanel>
                            <TextBlock Text="{Binding name}" FontSize="32" />
                            <TextBlock Text="{Binding phone}" FontSize="32" />
                        </StackPanel>
                    </StackPanel>
                </DataTemplate>
            </ListBox.ItemTemplate>
        </ListBox>

        <!-- Add -->
        <Button Content="Add Contact" FontSize="20" Padding="20 10 20 10" Margin="10 10 0 0" />
        </StackPanel>
    </Page>
```

MainPage.xaml.cs

```
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Navigation;

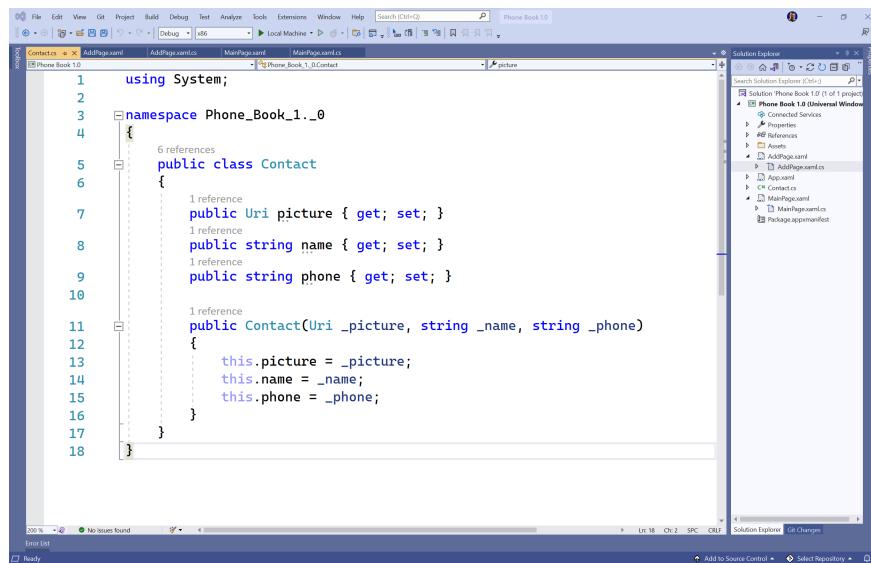
namespace Phone_Book_1._0
{
    /// <summary>
    /// Business Logic (BL): Phone Book 1.0
    /// </summary>
    public sealed partial class MainPage : Page
    {
        // Constructor
        public MainPage()
        {
            this.InitializeComponent();
            boxContacts.ItemsSource = App.contacts;
        }

        // Button Click Event Handler
        private void Button_Click(object sender, RoutedEventArgs e)
        {
            this.Frame.Navigate(typeof(AddPage));
        }

        // Navigation Event Handler
        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            if (e.Parameter is Contact)
            {
                App.contacts.Add(e.Parameter as Contact);
            }
        }
    }
}
```

Demo

Изглед в интегрираната среда за разработка Visual Studio по време на разработване на приложението:



Фиг. 32. Изглед от интегрираната среда за разработка по време на създаване на приложението

Стартирайте приложението от менюто: **Debug > Start Debugging** или като натиснете клавиш **F5**.



Фиг. 33. Универсално приложение тип телефонен указател съдържащо списък с контакти

Използвани изображения за потребителски профили

1. Мъж: <https://icons-for-free.com/iconfiles/png/512/business+costume+male+man+office+user+icon-1320196264882354682.png>
2. Жена: <https://icons-for-free.com/iconfiles/png/512/female+person+user+woman+young+icon-1320196266256009072.png>

Clicker Mania 1.0

Използвайки интегрираната среда за разработка Visual Studio и езика за програмиране C# ще разработим универсално приложение отчитащо броя кликове на потребителя.

Start

1. Стаптирайте интегрираната среда за разработка **Visual Studio**.
2. Създайте нов проект **Visual C# > Windows Universal > Blank App (Universal Windows)**.
3. За име на проекта запишете: **Clicker Mania 1.0**.

MainPage.xaml

Файлът **MainPage.xaml** съдържа изходния код от дизайна на потребителския интерфейс на разработваното приложение и се пише на езика XAML.
Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

```
<Page
    x:Class="Clicker_Mania_1._0.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:Clicker_Mania_1._0"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <!-- User Interface (UI): Clicker Mania 1.0 -->
    <StackPanel Background="Orange" Padding="50">

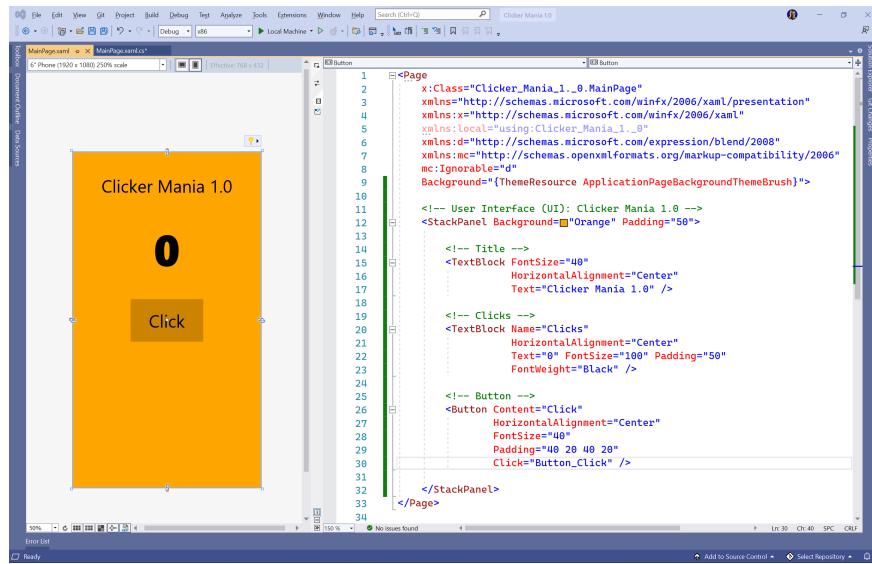
        <!-- Title -->
        <TextBlock FontSize="40"
            HorizontalAlignment="Center"
            Text="Clicker Mania 1.0" />

        <!-- Clicks -->
        <TextBlock Name="Clicks"
            HorizontalAlignment="Center"
            Text="0" FontSize="100" Padding="50"
            FontWeight="Black" />

        <!-- Button -->
        <Button Content="Click"
            HorizontalAlignment="Center"
            FontSize="40"
            Padding="40 20 40 20"
            Click="Button_Click" />

    </StackPanel>
</Page>
```

Изглед от дизайна на потребителският интерфейс (XAML) в интегрираната среда за разработка Visual Studio по време на разработване на приложението:



Фиг. 34. Изглед от дизайна на потребителският интерфейс

MainPage.xaml.cs

Файлът **MainPage.xaml.cs** съдържа изходния код от бизнес логиката на разработваното приложение и се пише на програмният език C#. Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

```

using System;
using System.IO;
using System.Text;
using Windows.Storage;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;

namespace Clicker_Mania_1._0
{
    /// <summary>
    /// Business Logic (BL): Clicker Mania 1.0
    /// </summary>
    public sealed partial class MainPage : Page
    {
        // Counter
        private int counter = 0;

        // Constructor
        public MainPage()
        {
            this.InitializeComponent();
            Read();
        }

        // Button Click Event Handler
        private void Button_Click(object sender, RoutedEventArgs e)
        {
            counter = counter + 1;
            Clicks.Text = counter.ToString();
            Save(Clicks.Text);
        }

        // Save to File
        private async void Save(string content)
        {
            try
            {
                StorageFolder storage = ApplicationData.Current.LocalFolder;
                byte[] bytes = Encoding.UTF8.GetBytes(content.ToArray());
                var file = await storage.CreateFileAsync("clicker.txt", CreationCollisionOption.ReplaceExisting);
                using (var stream = await file.OpenStreamForWriteAsync())
                {
                    stream.Write(bytes, 0, bytes.Length);
                }
            }
            catch
            {

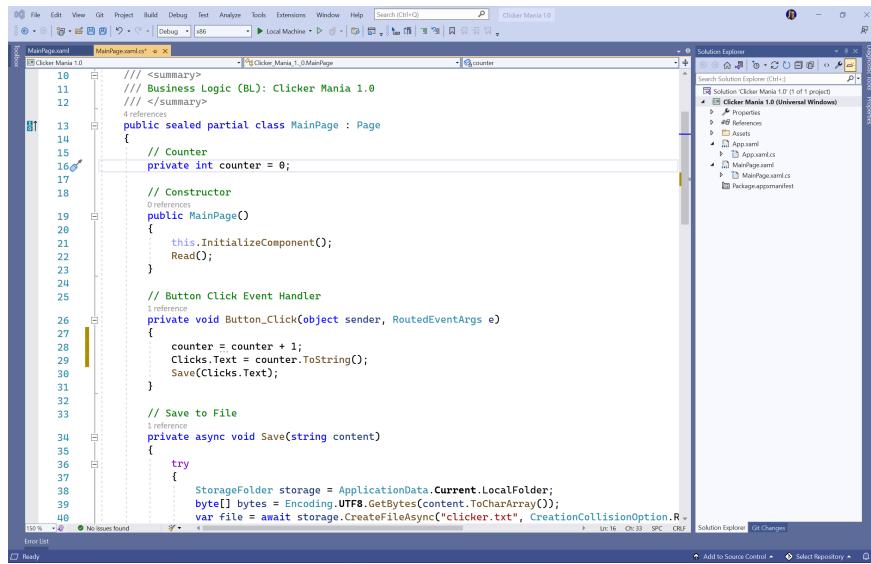
```

```
// On Error
}

}

// Read from File
private async void Read()
{
    try
    {
        StorageFolder storage = ApplicationData.Current.LocalFolder;
        StorageFile file = await storage.GetFileAsync("clicker.txt");
        if (file == null)
        {
            file = await storage.CreateFileAsync("clicker.txt");
        }
        else
        {
            Stream stream = await file.OpenStreamForReadAsync();
            StreamReader reader = new StreamReader(stream);
            Clicks.Text = reader.ReadToEnd();
            if (Clicks.Text == "")
            {
                Clicks.Text = "0";
                counter = 0;
            }
            else counter = int.Parse(Clicks.Text);
        }
    }
    catch
    {
        // On Error
    }
}
}
```

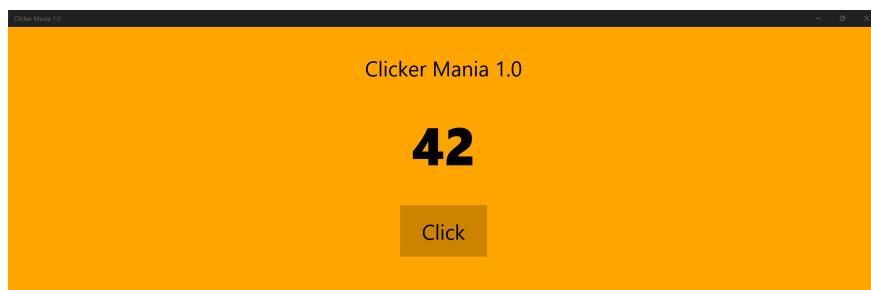
Изглед от бизнес логиката (C#) в интегрираната среда за разработка Visual Studio по време на разработване на приложението:



Фиг. 35. Изглед от бизнес логиката на разработваното приложение

Demo

Стартирайте приложението от менюто: **Debug > Start Debugging** или като натиснете клавиши **F5**.



Фиг.36. Универсалано приложение отчитащо броя кликове на потребителя.

Clicker Mania 2.0

Използвайки интегрираната среда за разработка Visual Studio и езика за програмиране C# ще разработим универсално приложение отчитащо броя кликове на потребителя за определено време.

Start

1. Стаптирайте интегрираната среда за разработка **Visual Studio**.
2. Създайте нов проект **Visual C# > Windows Universal > Blank App (Universal Windows)**.
3. За име на проекта запишете: **Clicker Mania 2.0**.

MainPage.xaml

Файлът **MainPage.xaml** съдържа изходния код от дизайна на потребителския интерфейс на разработваното приложение и се пише на езика XAML.
Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

```
<Page
    x:Class="Clicker_Mania_2._0.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:Clicker_Mania_2._0"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <!-- User Interface (UI): Clicker Mania 2.0 -->
    <StackPanel Background="Lime" Padding="30">

        <!-- Title -->
        <TextBlock FontSize="30" Padding="10" HorizontalAlignment="Center" Text="Clicker Mania 2.0" />

        <!-- Timer -->
        <TextBlock Text="Timer" FontSize="23" HorizontalAlignment="Center" />
        <TextBlock Name="Timer" Foreground="White" Text="0" FontSize="60" HorizontalAlignment="Center" />

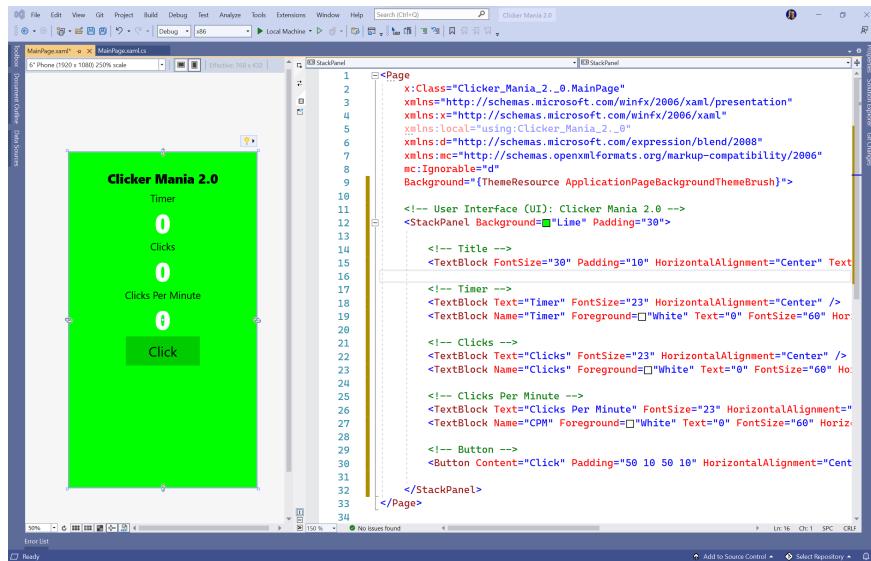
        <!-- Clicks -->
        <TextBlock Text="Clicks" FontSize="23" HorizontalAlignment="Center" />
        <TextBlock Name="Clicks" Foreground="White" Text="0" FontSize="60" HorizontalAlignment="Center" />

        <!-- Clicks Per Minute -->
        <TextBlock Text="Clicks Per Minute" FontSize="23" HorizontalAlignment="Center" />
        <TextBlock Name="CPM" Foreground="White" Text="0" FontSize="60" HorizontalAlignment="Center" />

        <!-- Button -->
        <Button Content="Click" Padding="50 10 50 10" HorizontalAlignment="Center" />

    </StackPanel>
</Page>
```

Изглед от дизайна на потребителският интерфейс (XAML) в интегрираната среда за разработка Visual Studio по време на разработване на приложението:



Фиг. 37. Изглед от дизайна на потребителският интерфейс

MainPage.xaml.cs

Файлът **MainPage.xaml.cs** съдържа изходния код от бизнес логиката на разработваното приложение и се пише на програмният език C#. Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

```
using System;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;

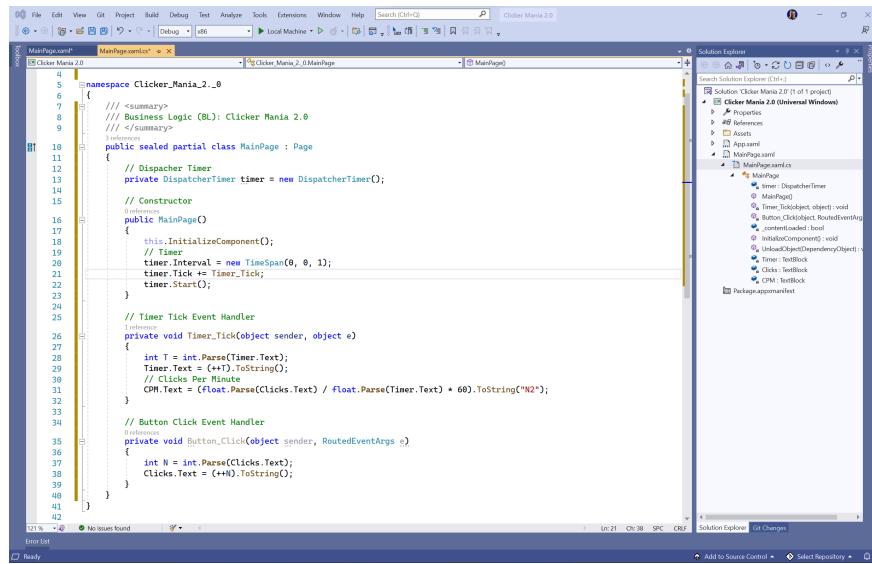
namespace Clicker_Mania_2._0
{
    /// <summary>
    /// Business Logic (BL): Clicker Mania 2.0
    /// </summary>
    public sealed partial class MainPage : Page
    {
        // Dispatcher Timer
        private DispatcherTimer timer = new DispatcherTimer();

        // Constructor
        public MainPage()
        {
            this.InitializeComponent();
            // Timer
            timer.Interval = new TimeSpan(0, 0, 1);
            timer.Tick += Timer_Tick;
            timer.Start();
        }

        // Timer Tick Event Handler
        private void Timer_Tick(object sender, object e)
        {
            int T = int.Parse(Timer.Text);
            Timer.Text = (++T).ToString();
            // clicks Per Minute
            CPM.Text = (float.Parse(Clicks.Text) / float.Parse(Timer.Text) * 60);
        }

        // Button Click Event Handler
        private void Button_Click(object sender, RoutedEventArgs e)
        {
            int N = int.Parse(Clicks.Text);
            Clicks.Text = (++N).ToString();
        }
    }
}
```

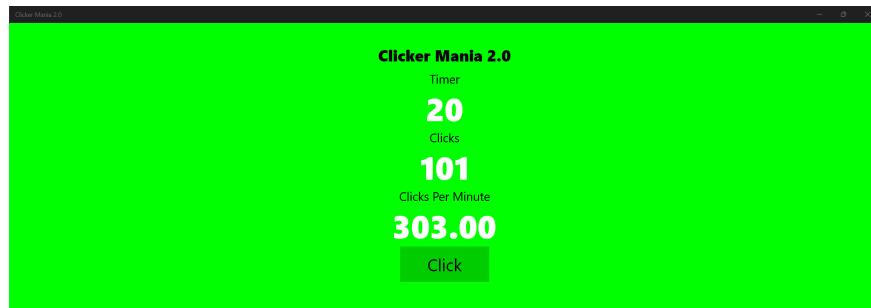
Изглед от бизнес логиката (C#) в интегрираната среда за разработка Visual Studio по време на разработване на приложението:



Фиг. 38. Изглед от бизнес логиката на разработваното приложение

Demo

Стартирайте приложението от менюто: **Debug > Start Debugging** или като натиснете клавиши **F5**.



Фиг.39 Универсално приложение отчитащо броя кликове на потребителя за определено време

HTML Downloader 1.0

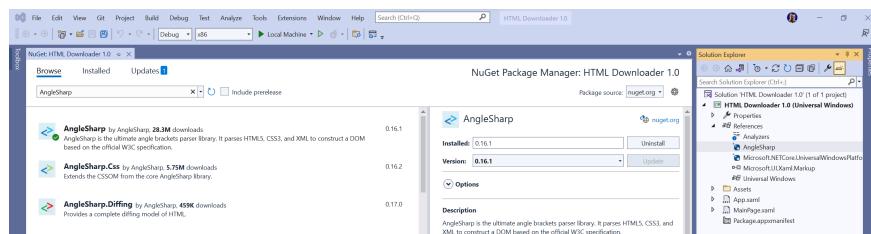
Използвайки интегрираната среда за разработка Visual Studio и езика за програмиране C# ще разработим универсално приложение за изтегляне HTML съдържанието на кода от Интернет страница.



Start

1. Стаптирайте интегрираната среда за разработка **Visual Studio**.
2. Създайте нов проект **Visual C# > Windows Universal > Blank App (Universal Windows)**.
3. За име на проекта запишете: **HTML Downloader 1.0**.

От менюто **Project > Manage NuGet Packages** потърсете и инсталирайте пакета **AngleSharp**, като е показано на фигурата:



Фиг. 40. Инсталация на допълнителен пакет към проекта

Допълнителни пакети към проект можете да инсталирате и алтернативно от менюто: **Tools > NuGet Package Manager > Package Manager Console**, като изпълните следната команда в конзолата:

```
PM> Install-Package AngleSharp -Version 0.16.1
```

MainPage.xaml

Файлът **MainPage.xaml** съдържа изходния код от дизайна на потребителския интерфейс на разработваното приложение и се пише на езика XAML.

Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

```

<Page
    x:Class="HTML_Downloader_1._0.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:HTML_Downloader_1._0"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <!-- User Interface (UI): HTML Downloader 1.0 -->
    <StackPanel Background="LightCoral" Padding="20">

        <!-- Title -->
        <TextBlock Text="HTML Downloader 1.0" FontSize="40" />

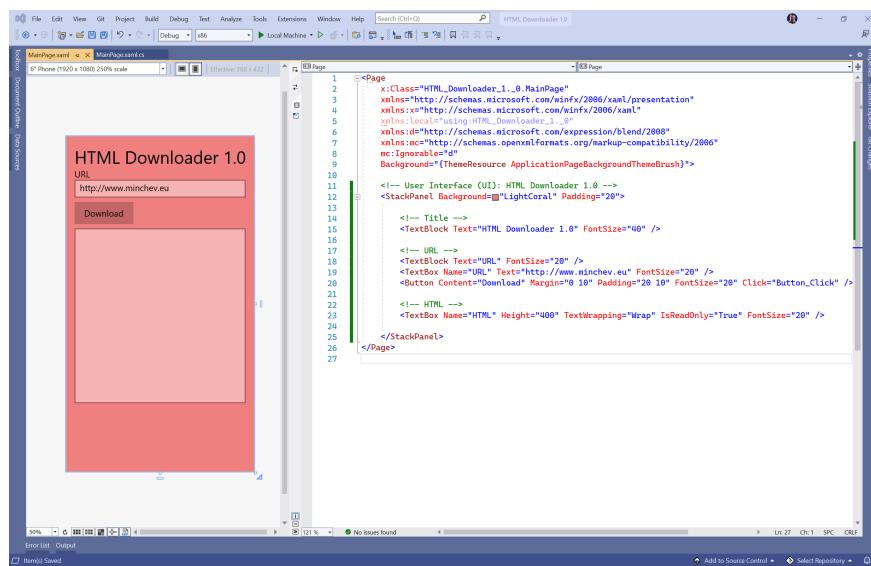
        <!-- URL -->
        <TextBlock Text="URL" FontSize="20" />
        <TextBox Name="URL" Text="http://www.minchev.eu" FontSize="20" />
        <Button Content="Download" Margin="0 10" Padding="20 10" FontSize="20" Click="Button_Click" />

        <!-- HTML -->
        <TextBox Name="HTML" Height="400" TextWrapping="Wrap" IsReadOnly="True" />

    </StackPanel>
</Page>

```

Изглед от дизайна на потребителският интерфейс (XAML) в интегрираната среда за разработка Visual Studio по време на разработване на приложението:



Фиг. 41. Изглед от дизайна на потребителският интерфейс

MainPage.xaml.cs

Файлът **MainPage.xaml.cs** съдържа изходния код от бизнес логиката на разработваното приложение и се пише на програмният език C#. Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

```
using System;
using System.Net.Http;
using System.Threading.Tasks;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using AngleSharp.Html.Parser;

namespace HTML_Downloader_1._0
{
    /// <summary>
    /// Business Logic (BL): HTML Downloader 1.0
    /// </summary>
    public sealed partial class MainPage : Page
    {
        // Constructor
        public MainPage()
        {
            this.InitializeComponent();
        }

        // Button Click Event Handler
        private async void Button_Click(object sender, RoutedEventArgs e)
        {
            string html = await Download(new Uri(URL.Text));
            var temp = new HtmlParser().ParseDocument(html);
            string text = temp.Body.TextContent;
            HTML.Text = text;
        }

        // Download Handler
        private async Task<string> Download(Uri link)
        {
            HttpClient client = new HttpClient();
            return await client.GetStringAsync(link);
        }
    }
}
```

Demo

Изглед от бизнес логиката (C#) в интегрираната среда за разработка Visual Studio по време на разработване на приложението:

The screenshot shows the Visual Studio interface with the code editor open for the file `MainPage.xaml.cs`. The code implements a `MainPage` class that handles a button click event to download HTML content from a URL and parse it using `AngleSharp.Html.Parser`. The Solution Explorer on the right shows the project structure for "HTML Downloader 1.0".

```

1  using System;
2  using System.Net.Http;
3  using System.Threading.Tasks;
4  using Windows.UI.Xaml;
5  using Windows.UI.Xaml.Controls;
6  using AngleSharp.Html.Parser;
7
8  namespace HTML_Downloader_1_0
9  {
10     /// <summary>
11     /// Business Logic (BL): HTML Downloader 1.0
12     </summary>
13     public sealed partial class MainPage : Page
14     {
15         // Constructor
16         public MainPage()
17         {
18             this.InitializeComponent();
19         }
20
21         // Button Click Event Handler
22         private async void Button_Click(object sender, RoutedEventArgs e)
23         {
24             string html = await Download(new Uri(URL.Text));
25             var temp = new HtmlParser().ParseDocument(html);
26             string text = temp.Body.TextContent;
27             HTML.Text = text;
28         }
29
30         // Download Handler
31         private async Task<string> Download(Uri link)
32         {
33             HttpClient client = new HttpClient();
34             return await client.GetStringAsync(link);
35         }
36     }
37 }
38

```

Фиг. 42. Изглед от бизнес логиката на разработваното приложение

Стартирайте приложението от менюто: **Debug > Start Debugging** или като натиснете клавиш **F5**.



Фиг.43 Универсално приложение за изтегляне HTML съдържанието на кода от Интернет страница

RSS Reader 1.0

Използвайки интегрираната среда за разработка Visual Studio и езика за програмиране C# ще разработим универсално приложение за четене на Интернет новинарски емисии от RSS източници.

Информация

RSS е софтуерен механизъм за обмен на новини между два сайта или между сайт и потребител. Представлява набор от формати за захранване с информация от световната Интернет мрежа.

- Източник: [Wikipedia](#)

Start

1. Стаптирайте интегрираната среда за разработка **Visual Studio**.
2. Създайте нов проект **Visual C# > Windows Universal > Blank App (Universal Windows)**.
3. За име на проекта запишете: **RSS Reader 1.0**.

Item.cs

Добавете нов клас **Item.cs**, който ще служи за съхранение на данни за всяка една новина от емисията.

```
namespace RSS_Reader_1._0
{
    public class Item
    {
        public string Title { get; set; }
        public string Link { get; set; }
        public string PublishedDate { get; set; }
    }
}
```

MainPage.xaml

Файлът **MainPage.xaml** съдържа изходния код от дизайна на потребителския интерфейс на разработваното приложение и се пише на езика XAML. Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.



The screenshot shows the XAML code for a mobile application's page. The code defines a `<Page>` element with various namespaces and properties. It includes sections for the UI (User Interface), such as a `<StackPanel>` containing a title and a download button, and a RSS feed viewer section with a `<ListView>` and `<DataTemplate>`. The code uses binding for the RSS feed items.

```
<Page
    x:Class="RSS_Reader_1._0.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:RSS_Reader_1._0"
    xmlns:d="http://schemas.microsoft.com/expressionblend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

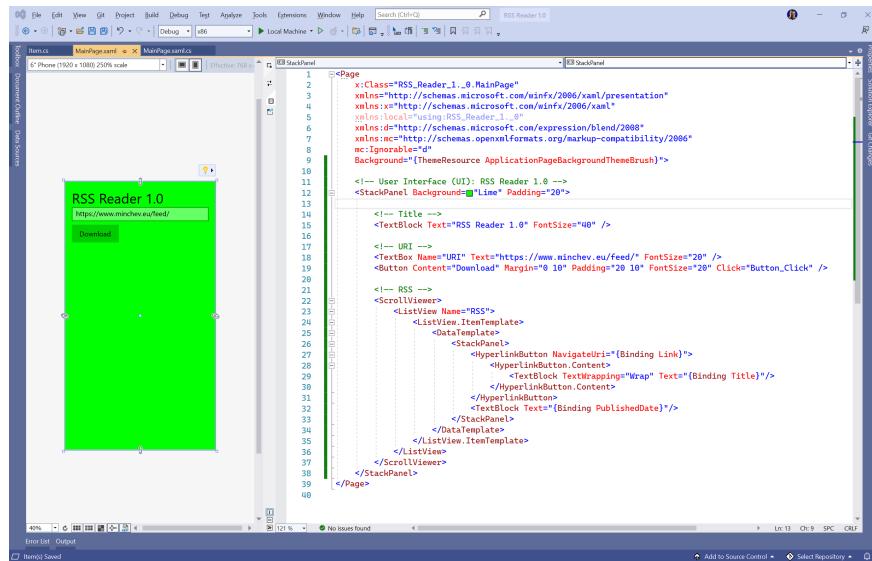
    <!-- User Interface (UI): RSS Reader 1.0 -->
    <StackPanel Background="Lime" Padding="20">

        <!-- Title -->
        <TextBlock Text="RSS Reader 1.0" FontSize="40" />

        <!-- URI -->
        <TextBox Name="URI" Text="https://www.minchev.eu/feed/" FontSize="20" />
        <Button Content="Download" Margin="0 10" Padding="20 10" FontSize="20" />

        <!-- RSS -->
        <ScrollViewer>
            <ListView Name="RSS">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <StackPanel>
                            <HyperlinkButton NavigateUri="{Binding Link}">
                                <HyperlinkButton.Content>
                                    <TextBlock TextWrapping="Wrap" Text="{Bind:}>
                                </HyperlinkButton.Content>
                            </HyperlinkButton>
                            <TextBlock Text="{Binding PublishedDate}"/>
                        </StackPanel>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </ScrollViewer>
    </StackPanel>
</Page>
```

Изглед от дизайна на потребителският интерфейс (XAML) в интегрираната среда за разработка Visual Studio по време на разработване на приложението:



Фиг. 44. Изглед от дизайна на потребителският интерфейс

MainPage.xaml.cs

Файлът **MainPage.xaml.cs** съдържа изходния код от бизнес логиката на разработваното приложение и се пише на програмният език C#. Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

```

using System;
using System.Collections.ObjectModel;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.Web.Syndication;

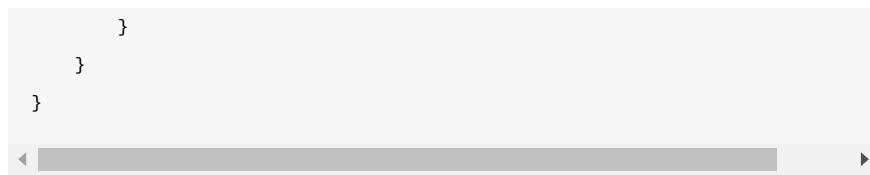
namespace RSS_Reader_1._0
{
    /// <summary>
    /// Business Logic(BL): RSS Reader 1.0
    /// </summary>
    public sealed partial class MainPage : Page
    {
        // View Model
        private ObservableCollection<Item> Items = new ObservableCollection<Item>();

        // Constructor
        public MainPage()
        {
            this.InitializeComponent();
            RSS.ItemsSource = Items;
        }

        // Button Click Event Handler
        private void Button_Click(object sender, RoutedEventArgs e)
        {
            Download();
        }

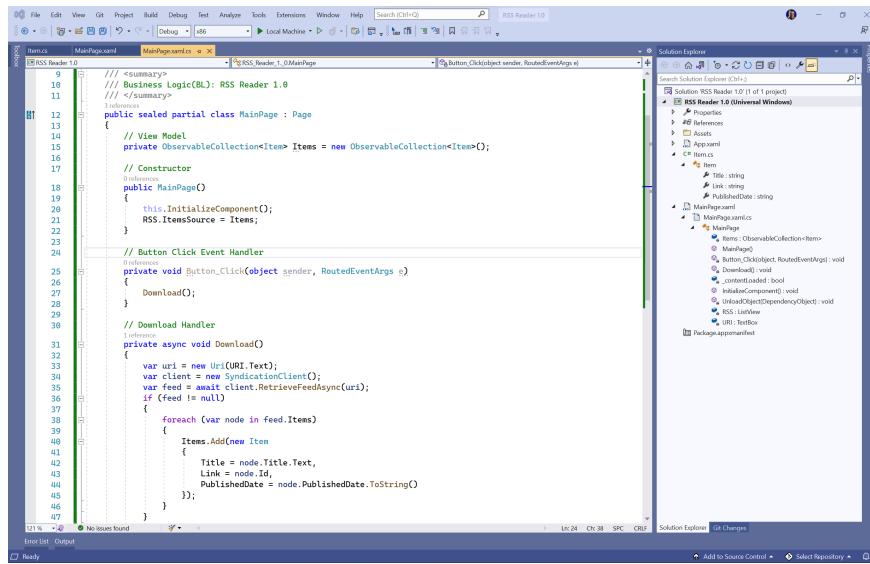
        // Download Handler
        private async void Download()
        {
            var uri = new Uri(URI.Text);
            var client = new SyndicationClient();
            var feed = await client.RetrieveFeedAsync(uri);
            if (feed != null)
            {
                foreach (var node in feed.Items)
                {
                    Items.Add(new Item
                    {
                        Title = node.Title.Text,
                        Link = node.Id,
                        PublishedDate = node.PublishedDate.ToString()
                    });
                }
            }
        }
    }
}

```



Demo

Изглед от бизнес логиката (C#) в интегрираната среда за разработка Visual Studio по време на разработване на приложението:



Фиг. 45. Изглед от бизнес логиката на разработваното приложение

Стартирайте приложението от менюто: **Debug > Start Debugging** или като натиснете клавиши **F5**.



Фиг.46 Универсално приложение за четене на Интернет новинарски емисии от RSS източници

JSON Reader 1.0

Използвайки интегрираната среда за разработка Visual Studio и езика за програмиране C# ще разработим универсално приложение за изтегляне на вицове за Чък Норис във формат [JSON](#).

Информация

JSON или JavaScript Object Notation, е текстово базиран отворен стандарт създаден за човешки четим обмен на данни. Произлиза от скриптовия език JavaScript, за да представя прости структури от данни и асоциативни масиви, наречени обекти.

- Източник: [Wikipedia](#)

Start

1. Стаптирайте интегрираната среда за разработка **Visual Studio**.
2. Създайте нов проект: **Visual C# > Windows Universal > Blank App (Universal Windows)**.
3. За име на проекта запишете: **JSON Reader 1.0**.

Добавете допълнителни пакети към проекта като инсталирате:

NewtownSoft.Json и **AngleSharp**, от менюто: **Tools > NuGet Package Manager > Package Manager Console**, като изпълните следните команди в конзолата:

```
PM> Install-Package Newtonsoft.Json -Version 13.0.1  
PM> Install-Package AngleSharp -Version 0.16.1
```

Root.cs

Добавете нов клас **Root.cs**, който ще служи за десериализиране на данните от консумираната услуга.

```
namespace JSON_Reader_1._0
{
    public class Root
    {
        public List<object> categories { get; set; }
        public string created_at { get; set; }
        public string icon_url { get; set; }
        public string id { get; set; }
        public string updated_at { get; set; }
        public string url { get; set; }
        public string value { get; set; }
    }
}
```

Бележка

1. Заредете и копирайте примерен JSON от:
<https://api.chucknorris.io/jokes/random>
2. Генерирайте C# класа на избрания JSON от:
<http://json2csharp.com/>

MainPage.xaml

Файлът **MainPage.xaml** съдържа изходния код от дизайна на потребителския интерфейс на разработваното приложение и се пише на езика XAML. Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

```

<Page
    x:Class="JSON_Reader_1._0.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:JSON_Reader_1._0"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <!-- User Interface (UI): JSON Reader 1.0 -->
    <StackPanel Background="LightSalmon" Padding="30">

        <!-- Title -->
        <TextBlock Text="JSON Reader 1.0" FontSize="40" Margin="10"/>

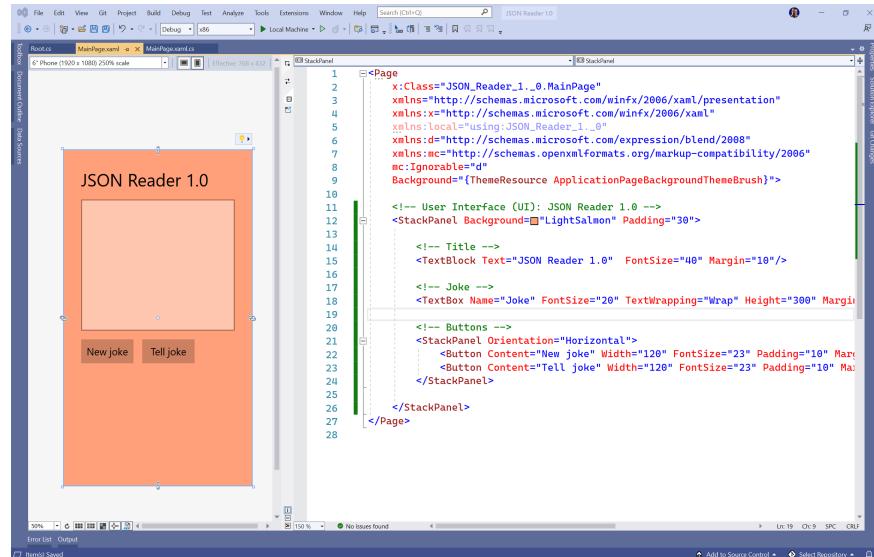
        <!-- Joke -->
        <TextBox Name="Joke" FontSize="20" TextWrapping="Wrap" Height="300" Margin="10" />

        <!-- Buttons -->
        <StackPanel Orientation="Horizontal">
            <Button Content="New joke" Width="120" FontSize="23" Padding="10" Margin="10" />
            <Button Content="Tell joke" Width="120" FontSize="23" Padding="10" Margin="10" />
        </StackPanel>
    </StackPanel>
</Page>

```



Изглед от дизайна на потребителския интерфейс (XAML) в интегрираната среда за разработка Visual Studio по време на разработване на приложението:



Фиг. 47. Изглед от дизайна на потребителският интерфейс

MainPage.xaml.cs

Файлът **MainPage.xaml.cs** съдържа изходния код от бизнес логиката на разработваното приложение и се пише на програмният език C#. Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

```

using System;
using System.Net.Http;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using AngleSharp.Html.Parser;
using Newtonsoft.Json;
using Windows.Media.SpeechSynthesis;

namespace JSON_Reader_1._0
{
    /// <summary>
    /// Business Logic (BL): JSON Reader 1.0
    /// </summary>
    public sealed partial class MainPage : Page
    {
        // Constructor
        public MainPage()
        {
            this.InitializeComponent();
        }

        // Button Get Joke Event Handler
        private async void Button_Get_Click(object sender, RoutedEventArgs e)
        {
            // Download JSON
            HttpClient client = new HttpClient();
            var json = await client.GetStringAsync(new Uri("https://api.chucknorris.io/jokes/random"));

            // Deserialize the JSON
            var joke = JsonConvert.DeserializeObject<Root>(json);

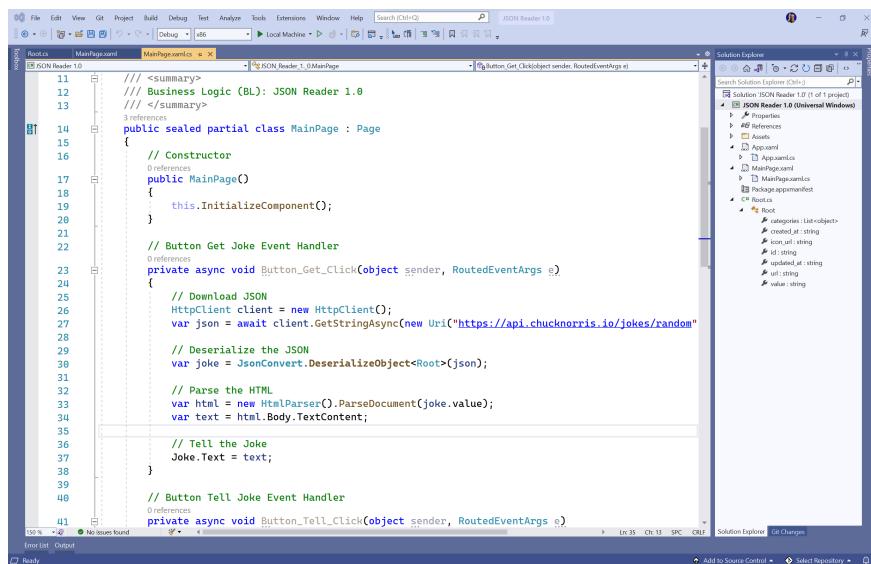
            // Parse the HTML
            var html = new HtmlParser().ParseDocument(joke.value);
            var text = html.Body.TextContent;

            // Tell the Joke
            Joke.Text = text;
        }

        // Button Tell Joke Event Handler
        private async void Button_Tell_Click(object sender, RoutedEventArgs e)
        {
            if (Joke.Text != "")
            {
                // The media object for controlling and playing audio.
                var mediaElement = new MediaElement();

```

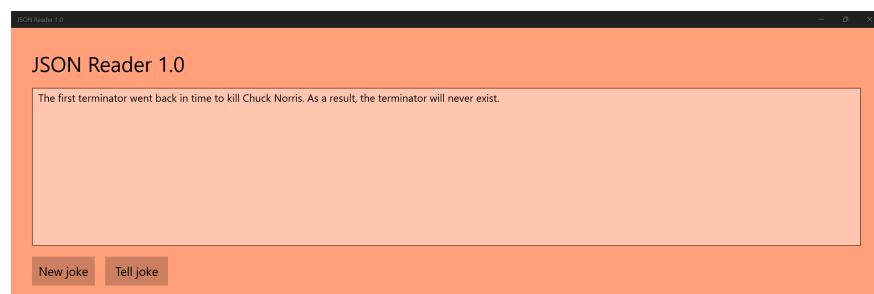
Изглед от бизнес логиката (C#) в интегрираната среда за разработка Visual Studio по време на разработване на приложението:



Фиг. 48. Изглед от бизнес логиката на разработваното приложение

Demo

Стартирайте приложението от менюто: **Debug > Start Debugging** или като натиснете клавиш **F5**.



Мобилни устройства

Фиг.49 Универсално приложение за изтегляне на вицове за Чък Норис

Мултиплатформени мобилни приложения

Информация

Настоящата глава от книгата въвежда читателя във вълнуващия свят на Microsoft технологиите за разработка на мултиплатформени мобилни приложения.

Съвет

Пожелавам Ви много забавление, провали и успехи използваки Microsoft технологиите за разработка на мултиплатформени мобилни приложения.

Забележка

Примерите в тази част са разработени използвайки следните версии на операционната система и интегрираната среда за разработка:

- Microsoft Windows 11 Version 10.0.22000.282
- Microsoft Visual Studio Community 2022 Version 17.0.0

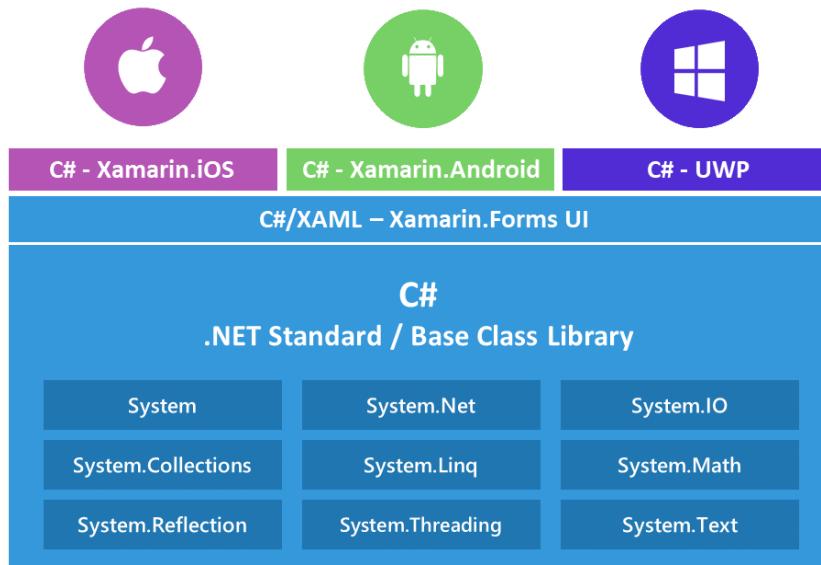
История

Основана Май 2011 година компанията Xamarin предоставя на програмистите единна платформа за разработка на мултиплатформени мобилни приложения. Екипа инженери автори на Mono, Mono for Android и Mono Touch обединиха своите усилия създавайки уникален продукт, който използвайки Microsoft технологиите C# и .NET, позволява разработване на Android, iOS и Windows приложения.



Фиг.50. Мултиплатформени мобилни приложения

През Февруари 2016 Microsoft обяви закупуването на компанията Xamarin. Технологичния гигант от Редмънд добави технологията към интегрираната среда за разработка Visual Studio. По данни от Април 2017 над 1.4 милиона разработчици от 120 страни използват Xamarin.



Фиг. 51 Платформено зависим и споделен изходен код

.NET MAUI

Xamarin.Forms еволюира в .NET MAUI през ноември 2021 когато Microsoft пуска .NET 6.

Мобилни устройства

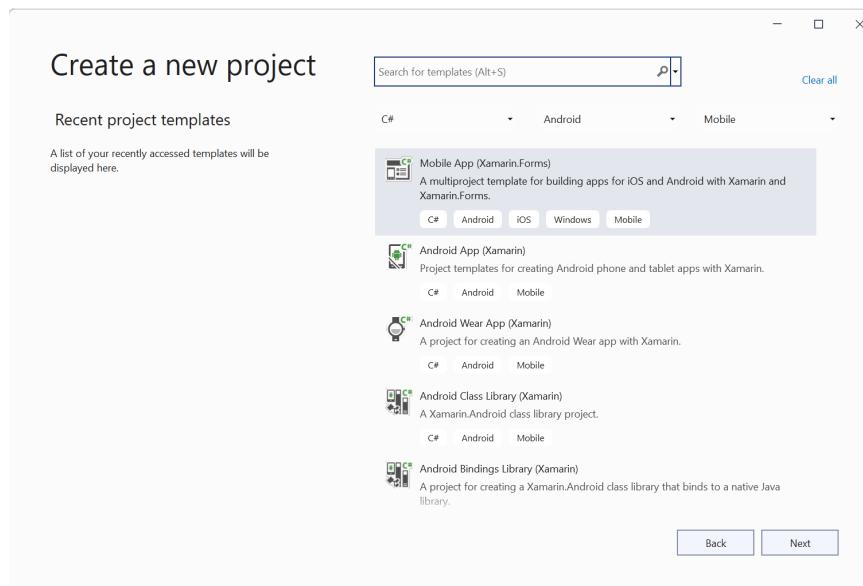
MAUI е съкращение от Multi-Platform App User Interface, което означава буквално "потребителски интерфейс на мулти-платформено приложение".

За повече информация за .NET MAUI вижте: [.NET Multi-Platform App User Interface Documentation](#).

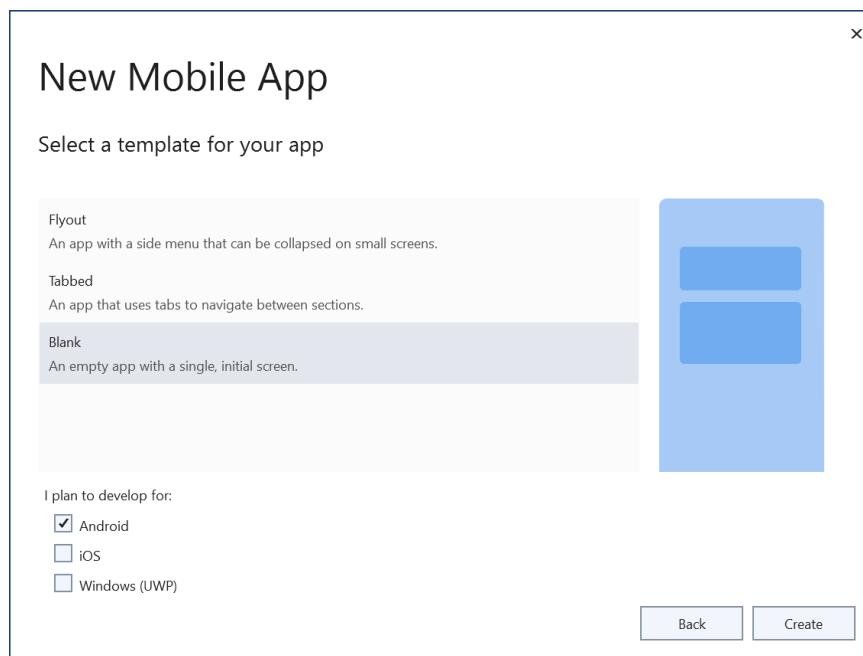
Приложения

В тази част са представени редица приложения демонстриращи разработка на мулти-платформени мобилни приложения.

Разработката на всяко приложение в интегрираната среда за разработка Visual Studio, започва със създаване на проект. Нов проект се създава от менюто посредством изпълняване на последователността: **File > New > Project**. Може да се използва и съкратената клавишка комбинация **Ctrl + Shift + N** за създаване на нов проект.



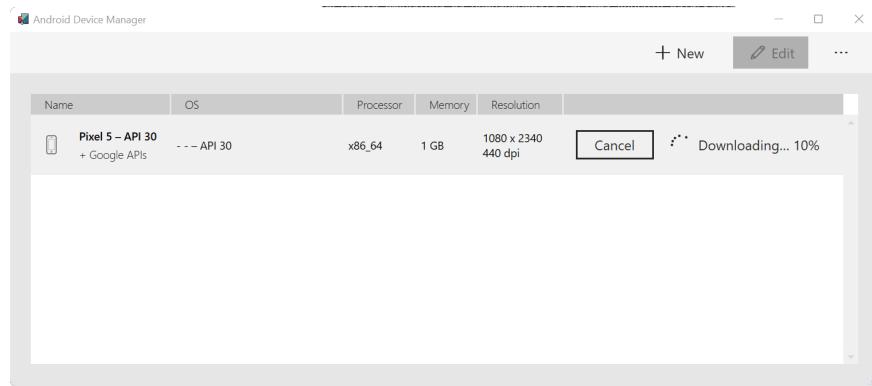
Фиг.52 Създаване на нов мулти-платформен проект



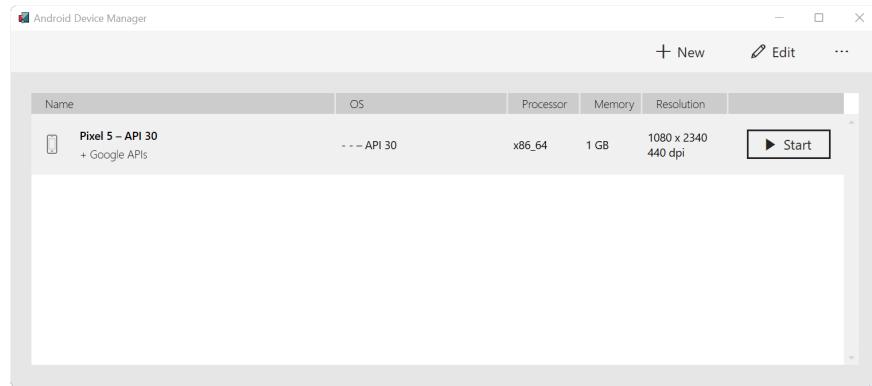
Фиг.53 Конфигуриране на новия мулти-платформен проект

Настройки

Меню **Tools > Android > Android Device Manager** добавете ново емулаторно устройство за тестване на разработваните приложения:



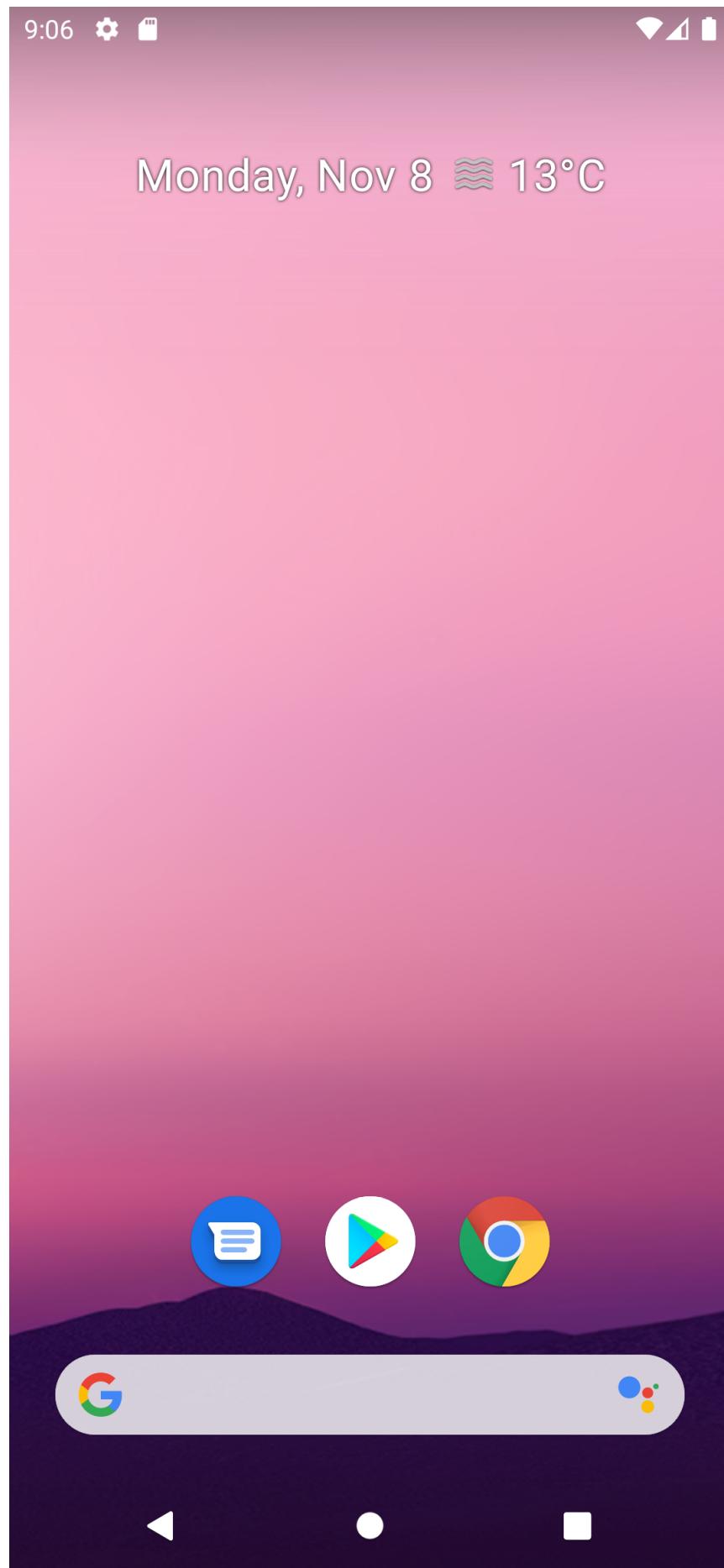
Фиг.54 Добавяне на ново емулаторно устройство за тестване на разработваните приложения.



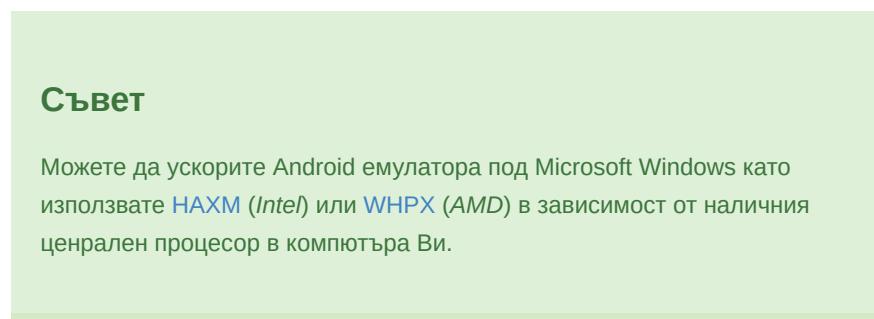
Фиг.55 Успешно е добавено ново емулаторно устройство за тестване на разработваните приложения.

Тествайте емулатора на операционната система Android натискайки бутона **Start**. Ако всичко е наред ще видите операционната система Android 11 (API 30), както е показано на фигурата по-долу:

Мобилни устройства



Фиг. 56. Android 11 (API 30)



Suma 2.0

Използвайки интегрираната среда за разработка Visual Studio и езика за програмиране C# ще разработим мулти-платформено мобилно приложение за намиране сумата на две числа.

Start

1. Стаптирайте интегрираната среда за разработка **Visual Studio**.
2. Създайте нов проект: **Visual C# > Cross-Platform > Mobile App (Xamarin.Forms)**.
3. За име на проекта запишете: **Suma 2.0**.

MainPage.xaml

Файлът **MainPage.xaml** съдържа изходния код от дизайна на потребителския интерфейс на разработваното приложение и се пише на езика XAML. Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="Suma_2._0.MainPage">

    <!-- User Interface (UI): Suma 2.0 -->
    <StackLayout Padding="20">

        <!-- Title -->
        <Label Text="Сума 2.0" FontSize="Large" />

        <!-- A -->
        <Label Text="A=" />
        <Entry x:Name="boxA" Keyboard="Numeric" />

        <!-- B -->
        <Label Text="B=" />
        <Entry x:Name="boxB" Keyboard="Numeric" />

        <!-- A+B -->
        <Label Text="A+B=" />
        <Entry x:Name="boxAB" Keyboard="Numeric" />

        <!-- Button -->
        <Button Text="Сум" Clicked="OnButtonClicked" />

    </StackLayout>
</ContentPage>
```

MainPage.xaml.cs

Файльт **MainPage.xaml.cs** съдържа изходния код от бизнес логиката на разработваното приложение и се пише на програмният език C#. Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

```

using System;
using Xamarin.Forms;

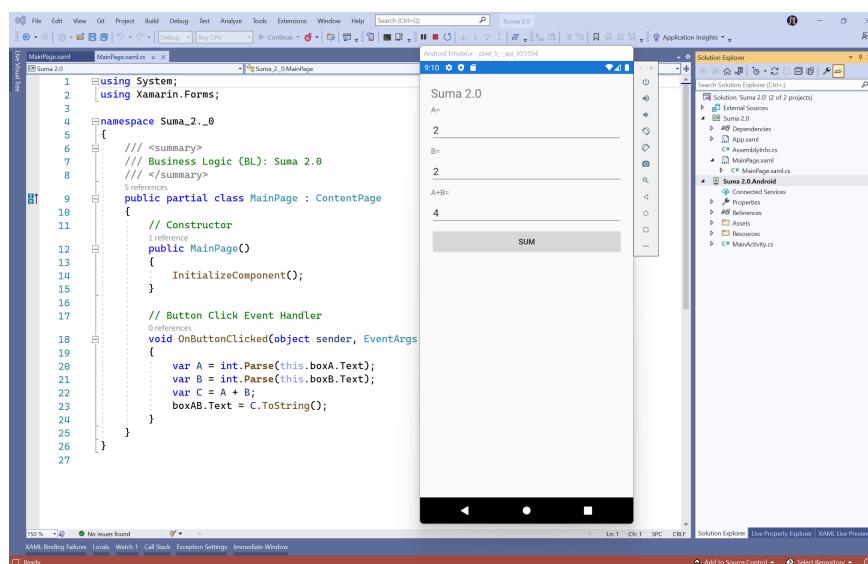
namespace Suma_2._0
{
    /// <summary>
    /// Business Logic (BL): Suma 2.0
    /// </summary>
    public partial class MainPage : ContentPage
    {
        // Constructor
        public MainPage()
        {
            InitializeComponent();
        }

        // Button Click Event Handler
        void OnButtonClicked(object sender, EventArgs args)
        {
            var A = int.Parse(this.boxA.Text);
            var B = int.Parse(this.boxB.Text);
            var C = A + B;
            boxAB.Text = C.ToString();
        }
    }
}

```

Demo

Стартирайте приложението от менюто: **Debug > Start Debugging** или като натиснете клавиш **F5**.



*Фиг.57 Тестване на мулти-платформено мобилно приложение за
намиране сумата на две числа - Android Emulator 11 (API 30)*

Fibonacci 2.0

Използвайки интегрираната среда за разработка Visual Studio и езика за програмиране C# ще разработим мулти-платформено мобилно приложение за генериране на числата от редицата на Фиbonacci.

Информация

Числата на Фибоначи в математиката образуват редица, която се дефинира рекурсивно по следния начин: започва се с 0 и 1, а всеки следващ член на редицата се получава като сума на предходните два. Първите числа на Фибоначи са: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

- Източник: [Wikipedia](#)

Start

1. Стаптирайте интегрираната среда за разработка **Visual Studio**.
2. Създайте нов проект: **Visual C# > Cross-Platform > Mobile App (Xamarin.Forms)**.
3. За име на проекта запишете: **Fibonacci 2.0**.

MainPage.xaml

Файлът **MainPage.xaml** съдържа изходния код от дизайна на потребителския интерфейс на разработваното приложение и се пише на езика XAML.

Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="Fibonacci_2._0.MainPage">

    <!-- User Interface (UI): Fibonacci 2.0 -->
    <StackLayout Padding="20">

        <!-- Title -->
        <Label Text="Fibonacci 2.0" FontSize="Large" />

        <!-- Limit -->
        <Label Text="Limit" />
        <Entry x:Name="boxLimit" Keyboard="Numeric" Text="1000" />
        <Button Text="Generate" Clicked="OnButtonClicked" />

        <!-- Numbers -->
        <ListView x:Name="boxNumbers" />

    </StackLayout>
</ContentPage>
```

MainPage.xaml.cs

Файлът **MainPage.xaml.cs** съдържа изходния код от бизнес логиката на разработваното приложение и се пише на програмният език C#. Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

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

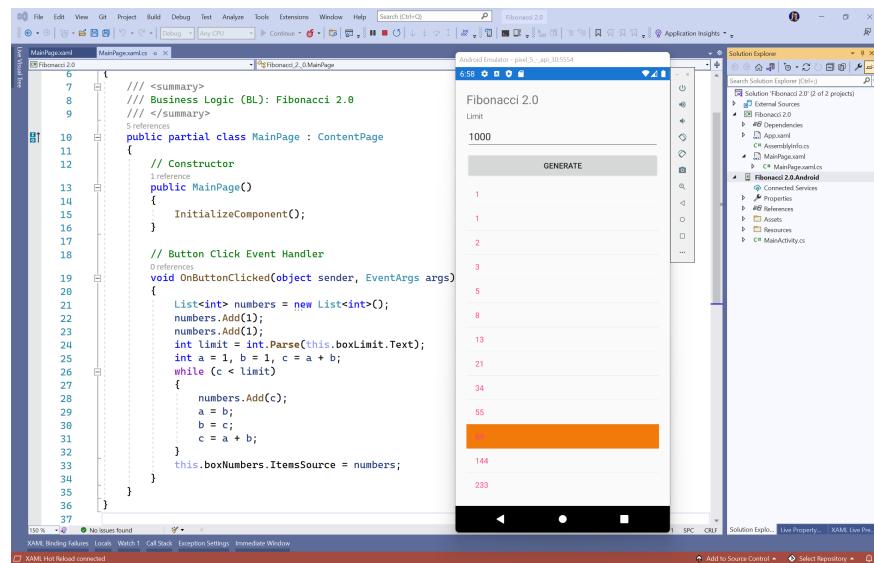
namespace Fibonacci_2._0
{
    /// <summary>
    /// Business Logic (BL): Fibonacci 2.0
    /// </summary>
    public partial class MainPage : ContentPage
    {
        // Constructor
        public MainPage()
        {
            InitializeComponent();
        }

        // Button Click Event Handler
        void OnButtonClicked(object sender, EventArgs args)
        {
            List<int> numbers = new List<int>();
            numbers.Add(1);
            numbers.Add(1);
            int limit = int.Parse(this.boxLimit.Text);
            int a = 1, b = 1, c = a + b;
            while (c < limit)
            {
                numbers.Add(c);
                a = b;
                b = c;
                c = a + b;
            }
            this.boxNumbers.ItemsSource = numbers;
        }
    }
}
```

Demo

Стартирайте приложението от менюто: **Debug > Start Debugging** или като натиснете клавиши **F5**.

Мобилни устройства



Фиг.58 Тестване на мултплатформено мобилно приложение за генериране на редицата от числата на Фибоначи - Android Emulator 11 (API 30)

Primes 2.0

Използвайки интегрираната среда за разработка Visual Studio и езика за програмиране C# ще разработим мулти-платформено мобилно приложение за генериране на редица от прости числа.

Информация

Просто число е естествено число, по-голямо от 1, което не е произведение на две по-малки естествени числа.

- Източник: [Wikipedia](#)

Start

1. Стаптирайте интегрираната среда за разработка **Visual Studio**.
2. Създайте нов проект: **Visual C# > Cross-Platform > Mobile App (Xamarin.Forms)**.
3. За име на проекта запишете: **Primes 2.0**.

MainPage.xaml

Файлът **MainPage.xaml** съдържа изходния код от дизайна на потребителския интерфейс на разработваното приложение и се пише на езика XAML.
Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="Primes_2._0.MainPage">

    <!-- User Interface (UI): Primes 2.0 -->
    <StackLayout Padding="20" BackgroundColor="Yellow">

        <!-- Title -->
        <Label Text="Primes 2.0" FontSize="Large" />

        <!-- Limit -->
        <Label Text="Limit" />
        <Entry x:Name="boxLimit" Keyboard="Numeric" Text="1000" />
        <Button Text="Generate" Clicked="OnButtonClicked" />

        <!-- Numbers -->
        <ListView x:Name="boxNumbers" />

    </StackLayout>
</ContentPage>
```

MainPage.xaml.cs

Файлът **MainPage.xaml.cs** съдържа изходния код от бизнес логиката на разработваното приложение и се пише на програмният език C#. Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

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

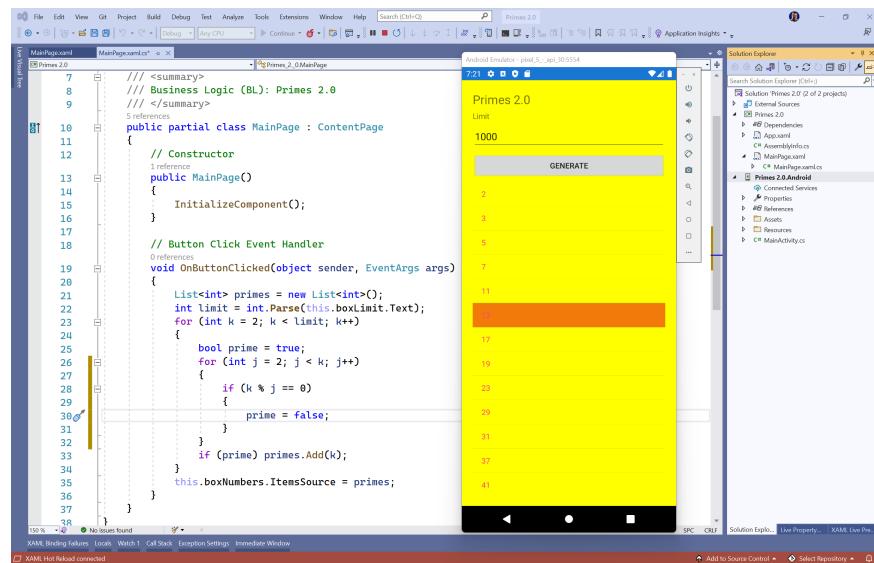
namespace Primes_2._0
{
    /// <summary>
    /// Business Logic (BL): Primes 2.0
    /// </summary>
    public partial class MainPage : ContentPage
    {
        // Constructor
        public MainPage()
        {
            InitializeComponent();
        }

        // Button Click Event Handler
        void OnButtonClicked(object sender, EventArgs args)
        {
            List<int> primes = new List<int>();
            int limit = int.Parse(this.boxLimit.Text);
            for (int k = 2; k < limit; k++)
            {
                bool prime = true;
                for (int j = 2; j < k; j++) if (k % j == 0) prime = false;
                if (prime) primes.Add(k);
            }
            this.boxNumbers.ItemsSource = primes;
        }
    }
}
```

Demo

Стартирайте приложението от менюто: **Debug > Start Debugging** или като натиснете клавиш **F5**.

Мобилни устройства



Фиг.59 Тестване на мултплатформено мобилно приложение за генериране на редица от прости числа - *Android Emulator 11 (API 30)*

Phone Book 2.0

Използвайки интегрираната среда за разработка Visual Studio и езика за програмиране C# ще разработим мулти-платформено мобилно приложение за телефонен указател съдържащ списък с контакти.

Start

1. Стаптирайте интегрираната среда за разработка **Visual Studio**.
2. Създайте нов проект: **Visual C# > Cross-Platform > Mobile App (Xamarin.Forms)**.
3. За име на проекта запишете: **Phone Book 2.0**.

Добавете допълнителни пакети към проекта като инсталирате:

NewtonSoft.Json от менюто: **Tools > NuGet Package Manager > Package Manager Console**, като изпълните следните команди в конзолата:

```
PM> Install-Package Newtonsoft.Json -Version 13.0.1
```

Contact.cs

Добавете нов клас **Contact.cs** съдържащ следният програмен фрагмент:

```
using System;

namespace Phone_Book_2._0.Model
{
    public class Contact
    {
        private Uri picture;

        public Uri Picture
        {
            get { return picture; }
            set { picture = value; }
        }

        private string name;

        public string Name
        {
            get { return name; }
            set { name = value; }
        }

        private string phone;

        public string Phone
        {
            get { return phone; }
            set { phone = value; }
        }

        public Contact(Uri picture, string name, string phone)
        {
            this.picture = picture;
            this.name = name;
            this.phone = phone;
        }
    }
}
```

ViewModel.cs

Добавете нов клас **ViewModel.cs** съдържащ следният програмен фрагмент:

```
using System;
using System.Collections;
using System.Collections.ObjectModel;
using Xamarin.Forms;

namespace Phone_Book_2._0.Model
{
    public static class ViewModel
    {
        public static ObservableCollection<Contact> Contacts;

        static ViewModel()
        {
            Contacts = new ObservableCollection<Contact>();

            BindingBase.EnableCollectionSynchronization(Contacts, null, Callback);
        }

        private static void Callback(IEnumerable collection, object context, Action accessMethod)
        {
            lock (collection)
            {
                accessMethod?.Invoke();
            }
        }
    }
}
```

MainPage.xaml

Файлът **MainPage.xaml** съдържа изходния код от дизайна на потребителския интерфейс на разработваното приложение и се пише на езика XAML.

Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="Phone_Book_2._0.MainPage">

    <StackLayout Padding="50">

        <Label Text="Phone Book 2.0" FontSize="Large" FontAttributes="Bold" />

        <Button Text="Add" Clicked="Button_Clicked" />

        <ListView x:Name="ListView">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <ImageCell Text="{Binding Name}"
                        Detail="{Binding Phone}"
                        ImageSource="{Binding Picture}" />
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>

    </StackLayout>

</ContentPage>
```

MainPage.xaml.cs

Файлът **MainPage.xaml.cs** съдържа изходния код от бизнес логиката на разработваното приложение и се пише на програмният език C#. Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

```
using Xamarin.Forms;
using Phone_Book_2._0.Model;

namespace Phone_Book_2._0
{
    public partial class MainPage : ContentPage
    {
        public MainPage(Contact contact = null)
        {
            InitializeComponent();

            this.ListView.ItemsSource = ViewModel.Contacts;

            if (contact is Contact)
            {
                ViewModel.Contacts.Add(contact);
            }
        }

        private async void Button_Clicked(object sender, System.EventArgs e)
        {
            await Navigation.PushModalAsync(new AddPage());
        }
    }
}
```

AddPage.xaml

Добавете нова страница **AddPage.xaml**. Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="Phone_Book_2._0.AddPage">

    <StackLayout Padding="50">

        <Label Text="Add" FontSize="Large" FontAttributes="Bold" />

        <Label Text="Picture" FontSize="Large" />
        <Entry x:Name="Picture" FontSize="Large" />

        <Label Text="Name" FontSize="Large" />
        <Entry x:Name="Name" FontSize="Large" />

        <Label Text="Phone" FontSize="Large" />
        <Entry x:Name="Phone" FontSize="Large" />

        <Button Text="Save" Clicked="Button_Clicked" />

    </StackLayout>

</ContentPage>
```

AppPage.xaml.cs

Отворете файла **AddPage.xaml.cs**. Копирайте (Ctrl+C) и поставете (Ctrl+V) програмния фрагмент даден по-долу във Вашето приложение.

```

using System;
using Xamarin.Forms;
using Phone_Book_2._0.Model;

namespace Phone_Book_2._0
{
    public partial class AddPage : ContentPage
    {
        public AddPage()
        {
            InitializeComponent();
        }

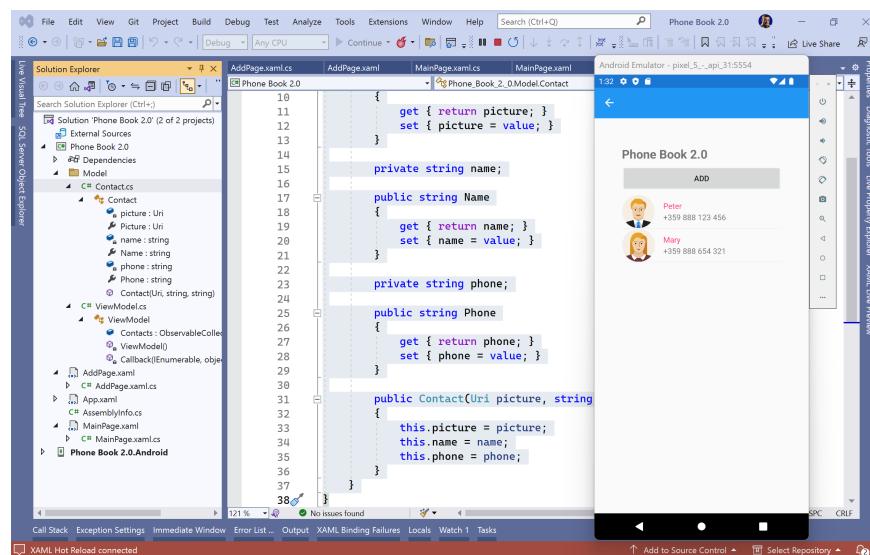
        private async void Button_Clicked(object sender, System.EventArgs e)
        {
            var contact = new Contact
            (
                picture: new Uri(this.Picture.Text),
                name: this.Name.Text,
                phone: this.Phone.Text
            );

            await Navigation.PushAsync(new MainPage(contact));
        }
    }
}

```

Demo

Стартирайте приложението от менюто: **Debug > Start Debugging** или като натиснете клавиш **F5**.



Фиг. 64. Демонстрация на работата на мулти-платформеното мобилно приложение за телефонен указател

Използвани изображения:

1. <https://shorturl.at/aiN17>
2. <https://shorturl.at/kALP3>

Clicker Mania 3.0

Използвайки интегрираната среда за разработка Visual Studio и езика за програмиране C# ще разработим мулти-платформено мобилно приложение отчитащо броя кликове на потребителя за определено време.

Start

1. Стаптирайте интегрираната среда за разработка **Visual Studio**.
2. Създайте нов проект: **Visual C# > Cross-Platform > Mobile App (Xamarin.Forms)**.
3. За име на проекта запишете: **Clicker Mania 3.0**.

MainPage.xaml

Файлът **MainPage.xaml** съдържа изходния код от дизайна на потребителския интерфейс на разработваното приложение и се пише на езика XAML.

Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="Clicker_Mania_3._0.MainPage">

    <!-- User Interface (UI): Clicker Mania 3.0 -->
    <StackLayout Padding="20" BackgroundColor="Lime">

        <!-- Title -->
        <Label Text="Clicker Mania 3.0" FontSize="Large" />

        <!-- Timer -->
        <Label Text="Timer" />
        <Entry x:Name="Timer" Text="0" />

        <!-- Clicks -->
        <Label Text="Clicks" />
        <Entry x:Name="Clicks" Text="0" />

        <!-- Clicks Per Minute -->
        <Label Text="Clicks Per Minute" />
        <Entry x:Name="CPM" Text="0" />

        <!-- Button -->
        <Button Text="Click" Clicked="OnButtonClicked" />

    </StackLayout>
</ContentPage>
```

MainPage.xaml.cs

Файльт **MainPage.xaml.cs** съдържа изходния код от бизнес логиката на разработваното приложение и се пише на програмният език C#. Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

```
using System;
using Xamarin.Forms;

namespace Clicker_Mania_3._0
{
    // Business Logic (BL): Clicker Mania 3.0
    public partial class MainPage : ContentPage
    {
        // Constructor
        public MainPage()
        {
            this.InitializeComponent();

            // Timer
            Device.StartTimer(TimeSpan.FromSeconds(1), TimerTick);
        }

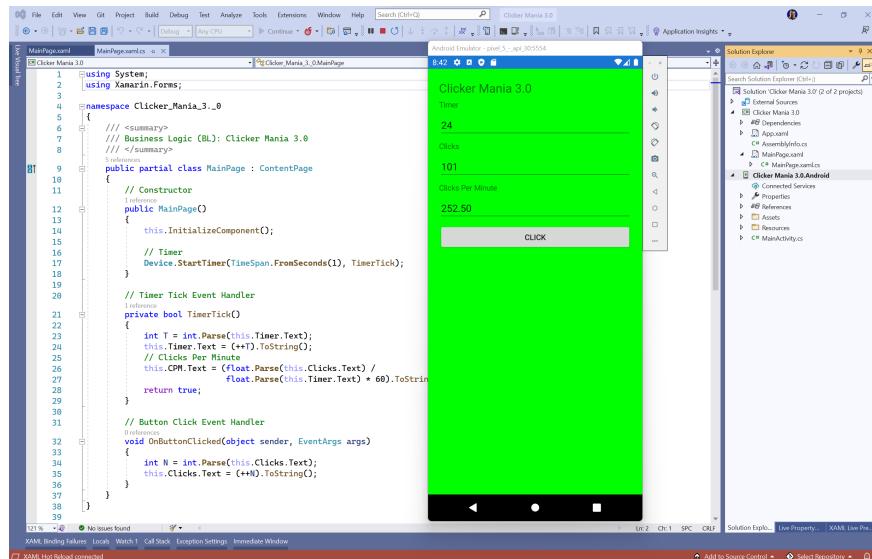
        // Timer Tick Event Handler
        private bool TimerTick()
        {
            int T = int.Parse(this.Timer.Text);
            this.Timer.Text = (++T).ToString();
            // clicks Per Minute
            this.CPM.Text = (float.Parse(this.Clicks.Text) /
                float.Parse(this.Timer.Text) * 60).ToString("N2");
            return true;
        }

        // Button Click Event Handler
        void OnButtonClicked(object sender, EventArgs args)
        {
            int N = int.Parse(this.Clicks.Text);
            this.Clicks.Text = (++N).ToString();
        }
    }
}
```

Demo

Стартирайте приложението от менюто: **Debug > Start Debugging** или като натиснете клавиши **F5**.

Мобилни устройства



Фиг.60 Тестване на мултплатформено мобилно приложение отчитащо броя кликове на потребителя за определено време - *Android Emulator 11 (API 30)*

HTML Downloader 2.0

Използвайки интегрираната среда за разработка Visual Studio и езика за програмиране C# ще разработим мулти-платформено мобилно приложение за изтегляне HTML съдържанието на кода от Интернет страница.

Информация

HTML е основният маркиращ език за описание и дизайн на уеб страници. HTML е стандарт в Интернет, а правилата се определят от международния консорциум W3C.

- Източник: [Wikipedia](#)

Start

1. Стаптирайте интегрираната среда за разработка **Visual Studio**.
2. Създайте нов проект: **Visual C# > Cross-Platform > Mobile App (Xamarin.Forms)**.
3. За име на проекта запишете: **HTML Downloader 2.0**.

Инсталирайте допълнителен пакет към приложението от менюто: **Tools > NuGet Package Manager > Package Manager Console**, като изпълните следната команда в конзолата:

```
PM> Install-Package AngleSharp -Version 0.16.1
```

MainPage.xaml

Файлът **MainPage.xaml** съдържа изходния код от дизайна на потребителския интерфейс на разработваното приложение и се пише на езика XAML.

Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="HTML_Downloader_2._0.MainPage">

    <!-- User Interface (UI): HTML Downloaders 2.0 -->
    <StackLayout Padding="20" BackgroundColor="LightCoral">

        <!-- Title -->
        <Label Text="HTML Downloader 2.0" FontSize="Large" />

        <!-- URL -->
        <Label Text="URL" FontSize="Large" />
        <Entry x:Name="URL" Text="https://www.minchev.eu" />
        <Button Text="Download" Clicked="OnButtonClicked" />

        <!-- HTML -->
        <ScrollView>
            <Label x:Name="HTML" />
        </ScrollView>

    </StackLayout>
</ContentPage>
```

MainPage.xaml.cs

Файлът **MainPage.xaml.cs** съдържа изходния код от бизнес логиката на разработваното приложение и се пише на програмният език C#. Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

```
using System;
using System.Net.Http;
using System.Threading.Tasks;
using Xamarin.Forms;
using AngleSharp.Html.Parser;

namespace HTML_Downloader_2._0
{
    /// <summary>
    /// Business Logic (BL): HTML Downloader 2.0
    /// </summary>
    public partial class MainPage : ContentPage
    {
        // Constructor
        public MainPage()
        {
            InitializeComponent();
        }

        // Button Click Event Handler
        private async void OnButtonClicked(object sender, EventArgs args)
        {
            // Get Html
            string html = await Download(new Uri(this.URL.Text));

            // Angle Sharp Html to Text Parser
            var temp = new HtmlParser().ParseDocument(html);
            string text = temp.Body.TextContent;

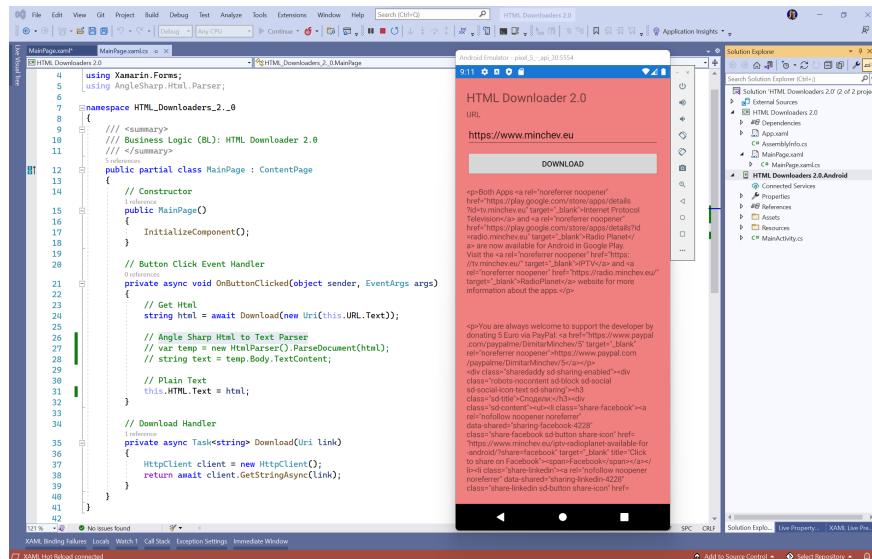
            // Plain Text
            this.HTML.Text = text;
        }

        // Download Handler
        private async Task<string> Download(Uri link)
        {
            HttpClient client = new HttpClient();
            return await client.GetStringAsync(link);
        }
    }
}
```

Demo

Стартирайте приложението от менюто: **Debug > Start Debugging** или като натиснете клавиши **F5**.

Мобилни устройства



Фиг.61 Тестване на мултплатформено мобилно приложение за изтегляне HTML съдържанието на кода от Интернет страница - Android Emulator 11 (API 30).

RSS Reader 2.0

Използвайки интегрираната среда за разработка Visual Studio и езика за програмиране C# ще разработим мулти-платформено мобилно приложение за четене на Интернет новинарски емисии от RSS източници.

Информация

RSS е софтуерен механизъм за обмен на новини между два сайта или между сайт и потребител. Представлява набор от формати за захранване с информация от световната Интернет мрежа.

- Извор: [Wikipedia](#)

Start

1. Стаптирайте интегрираната среда за разработка **Visual Studio**.
2. Създайте нов проект: **Visual C# > Cross-Platform > Mobile App (Xamarin.Forms)**.
3. За име на проекта запишете: **RSS Reader 2.0**.

FeedReader.cs

Добавете нов клас наречен **FeedReader.cs**, който ще служи за съхранение на данни от новинарската емисия.

```
using System.Linq;
using System.Xml.Linq;
using System.Collections.Generic;

namespace RSS_Reader_2._0
{
    /// <summary>
    /// RSS Feed Item
    /// </summary>
    public class FeedItem
    {
        public string Title { get; set; }
        public string Description { get; set; }
        public string Link { get; set; }
        public string PubDate { get; set; }
    }

    /// <summary>
    /// RSS Feed Reader
    /// </summary>
    public class FeedReader
    {
        public IEnumerable<FeedItem> ReadFeed(string url)
        {
            var feed = XDocument.Load(url);
            var posts = from item in feed.Descendants("item")
                        select new FeedItem
                        {
                            Title = item.Element("title").Value,
                            Description = item.Element("description").Value,
                            Link = item.Element("link").Value,
                            PubDate = item.Element("pubDate").Value
                        };
            return posts;
        }
    }
}
```

HyperlinkButton.cs

Добавете клас **HyperlinkButton.cs**, който ще служи за добавяне на хипервръзки към потребителския интерфейс на приложението, съгласно документацията на Microsoft: [Xamarin.Forms Hyperlinks Microsoft Docs](#)

```
using Xamarin.Forms;
using Xamarin.Essentials;

namespace RSS_Reader_2._0
{
    /// <summary>
    /// Hyperlink Button Control
    /// </summary>
    public class HyperlinkButton : TextCell
    {
        // Url Property
        public static readonly BindableProperty UrlProperty = BindableProperty
            .nameof\Url, typeof(string), typeof(HyperlinkBut
        public string Url
        {
            get { return (string)GetValue(UrlProperty); }
            set { SetValue(UrlProperty, value); }
        }

        // Constructor
        public HyperlinkButton()
        {
            Tapped += HyperlinkButton_Tapped;
        }

        // Tapped Event Handler
        private void HyperlinkButton_Tapped(object sender, System.EventArgs e)
        {
            // Launcher.OpenAsync is provided by Xamarin.Essentials.
            Command = new Command(async () => await Launcher.OpenAsync(Url));
        }
    }
}
```

MainPage.xaml

Файлът **MainPage.xaml** съдържа изходния код от дизайна на потребителския интерфейс на разработваното приложение и се пише на езика XAML. Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

```
<?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:RSS_Reader_2._0"
    x:Class="RSS_Reader_2._0.MainPage">

    <!-- User Interface (UI): RSS Reader 2.0 -->
    <StackLayout Padding="20" BackgroundColor="LightBlue">

        <!-- Title -->
        <Label Text="RSS Reader 2.0" FontSize="Large" />

        <!-- URI -->
        <Entry x:Name="URI" Text="https://www.minchev.eu/feed/" />
        <Button Text="Download" Clicked="Button_Clicked" />

        <!-- RSS -->
        <ListView x:Name="RSS">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <local:HyperlinkButton Text="{Binding Title}" Detail="{Binding Description}" />
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
    </StackLayout>
</ContentPage>
```

MainPage.xaml.cs

Файлът **MainPage.xaml.cs** съдържа изходния код от бизнес логиката на разработваното приложение и се пише на програмният език C#. Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

```
using System;
using System.Collections.ObjectModel;
using Xamarin.Forms;

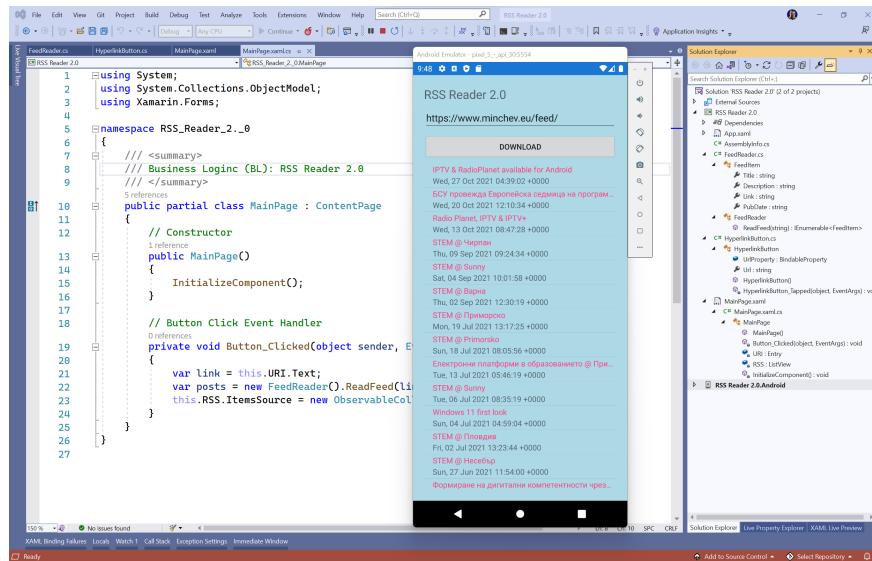
namespace RSS_Reader_2._0
{
    /// <summary>
    /// Business Logic (BL): RSS Reader 2.0
    /// </summary>
    public partial class MainPage : ContentPage
    {
        // Constructor
        public MainPage()
        {
            InitializeComponent();
        }

        // Button Click Event Handler
        private void Button_Clicked(object sender, EventArgs e)
        {
            RSS.ItemsSource = new ObservableCollection<FeedItem>
            (
                new FeedReader().ReadFeed(URI.Text)
            );
        }
    }
}
```

Demo

Стартирайте приложението от менюто: **Debug > Start Debugging** или като натиснете клавиши **F5**.

Мобилни устройства



Фиг.62 Тестване на на мултиплатформено мобилно приложение за четене на Интернет новинарски емисии от RSS източници- Android Emulator 11 (API 30).

JSON Reader 2.0

Използвайки интегрираната среда за разработка Visual Studio и езика за програмиране C# ще разработим мулти-платформено мобилно приложение за изтегляне на вицове за Чък Норис във формат [JSON](#).

Информация

JSON или JavaScript Object Notation, е текстово базиран отворен стандарт създаден за човешки четим обмен на данни. Произлиза от скриптовия език JavaScript, за да представя прости структури от данни и асоциативни масиви, наречени обекти.

- Източник: [Wikipedia](#)

Start

1. Стаптирайте интегрираната среда за разработка **Visual Studio**.
2. Създайте нов проект: **Visual C# > Cross-Platform > Mobile App (Xamarin.Forms)**.
3. За име на проекта запишете: **JSON Reader 2.0**.

Добавете допълнителни пакети към проекта като инсталирате: **NewtonSoft.Json** от менюто: **Tools > NuGet Package Manager > Package Manager Console**, като изпълните следните команди в конзолата:

```
PM> Install-Package Newtonsoft.Json -Version 13.0.1
```

Root.cs

Добавете нов клас **Root.cs**, който ще служи за десериализиране на данните от консумираната услуга.

```
using System.Collections.Generic;
namespace JSON_Reader_2._0
{
    public class Root
    {
        public List<object> categories { get; set; }

        public string created_at { get; set; }

        public string icon_url { get; set; }

        public string id { get; set; }

        public string updated_at { get; set; }

        public string url { get; set; }

        public string value { get; set; }
    }
}
```

Бележка

1. Заредете и копирайте примерен JSON от:
<https://api.chucknorris.io/jokes/random>
2. Генерирайте C# класа на избрания JSON от:
<http://json2csharp.com/>

MainPage.xaml

Файлът **MainPage.xaml** съдържа изходния код от дизайна на потребителския интерфейс на разработваното приложение и се пише на езика XAML.

Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="JSON_Reader_2._0.MainPage">

    <!-- JSON Reader 2.0 User Interface -->
    <StackLayout Padding="50">

        <!-- Title -->
        <Label Text="JSON Reader 2.0" FontSize="Large" FontAttributes="Bold" />

        <!-- Buttons -->
        <StackLayout Orientation="Horizontal">
            <Button Text="New Joke" Clicked="NewJokeButtonClickedEventHandler" />
            <Button Text="Tell Joke" Clicked="TellJokeButtonClickedEventHandler" />
        </StackLayout>

        <!-- Joke -->
        <Label x:Name="JOKE" FontSize="Large" />

    </StackLayout>

</ContentPage>
```

MainPage.xaml.cs

Файлът **MainPage.xaml.cs** съдържа изходния код от бизнес логиката на разработваното приложение и се пише на програмният език C#. Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

```

using System;
using System.Net.Http;
using Xamarin.Forms;
using Xamarin.Essentials;
using Newtonsoft.Json;

namespace JSON_Reader_2._0
{
    public partial class MainPage : ContentPage
    {
        /// <summary>
        /// Constructor
        /// </summary>
        public MainPage()
        {
            InitializeComponent();
        }

        /// <summary>
        /// New Joke Button Clicked Event Handler
        /// </summary>
        private async void NewJokeButtonClickedEventHandler(object sender, EventArgs e)
        {
            // 1. Http Client
            HttpClient client = new HttpClient();

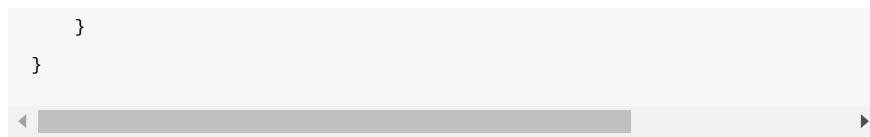
            // 2. Http Request to Receive JSON Response
            string json = await client.GetStringAsync(new Uri("https://api.chucknorris.io/jokes/random"));

            // 3. Deserialize JSON to Object
            Joke joke = JsonConvert.DeserializeObject<Joke>(json);

            // 4. Show the Joke in the UI
            JOKE.Text = joke.value;
        }

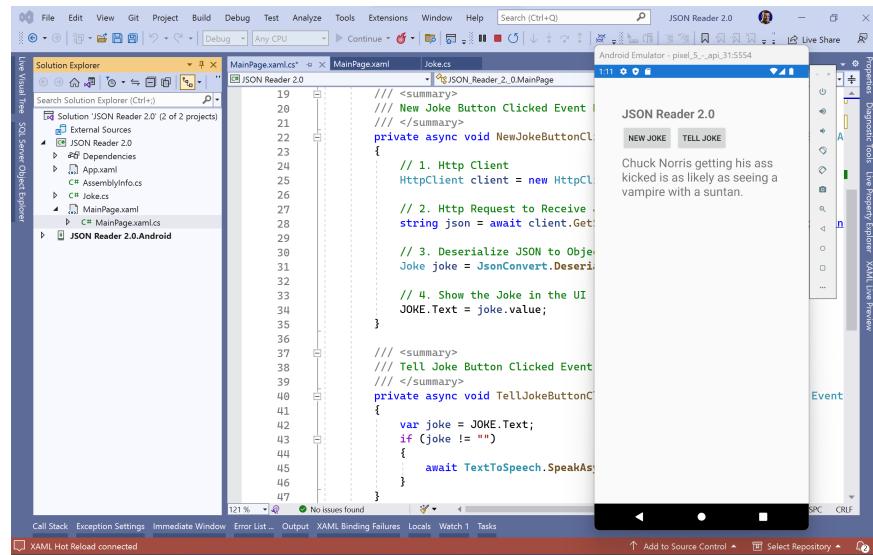
        /// <summary>
        /// Tell Joke Button Clicked Event Handler
        /// </summary>
        private async void TellJokeButtonClickedEventHandler(object sender, EventArgs e)
        {
            var joke = JOKE.Text;
            if (joke != "")
            {
                await TextToSpeech.SpeakAsync(joke);
            }
        }
    }
}

```



Demo

Стартирайте приложението от менюто: **Debug > Start Debugging** или като натиснете клавиш **F5**.



Фиг. 63. Демонстрация на работата на мулти-платформеното мобилно приложение за изтегляне на вицове за Чък Норис

Заключение

- [Бележка](#)
- [Книга](#)
- [Автор](#)
- [Лиценз](#)
- [Формат](#)

Бележка

Информация

Как да конвертирате настоящата GitBook електронна книга в PDF, EPUB и/или MOBI файлов формат?

Windows

1. Инсталирайте следните програмни продукти:

- [node.js](#)
- [Git](#)
- [Calibre](#)

2. Стаптирайте командния промпт като администратор и клонирайте електронното хранилище на книгата:

```
git clone https://github.com/dimitarminchev/DCPA.git
```

Резултат:

```
Cloning into 'DCPA'...
remote: Enumerating objects: 700, done.
remote: Counting objects: 100% (107/107), done.
remote: Compressing objects: 100% (85/85), done.
remote: Total 700 (delta 70), reused 36 (delta 22), pack-reused 593
Receiving objects: 100% (700/700), 204.77 MiB | 9.35 MiB/s, done.
Resolving deltas: 100% (410/410), done.
```

3. Влезте в току що клонираното електронно хранилище на книгата:

```
cd DCPA
```

4. Инсталирайте необходимите модули, като изпълнете следните команди:

```
$env:Path += 'C:\Program Files\Calibre2\'  
npm install -g ebook-convert  
npm install -g gitbook-cli  
gitbook install
```

5. Възможно е да получите подобна грешка:

```
C:\Users\mitko\AppData\Roaming\npm\node_modules\gitbook-cli\node_modules\npm\node  
if (cb) cb.apply(this, arguments)  
^  
TypeError: cb.apply is not a function  
at C:\Users\mitko\AppData\Roaming\npm\node_modules\gitbook-cli\node_modules\npm\node  
at FSReqCallback.oncomplete (node:fs:211:5)
```

За да поправите тази грешка:

```
cd C:\Users\mitko\AppData\Roaming\npm\node_modules\gitbook-cli\node_modules\npm  
npm i graceful-fs@4.1.4 --save
```

6. Върнете се в командният промпт и довършете инсталацията:

```
gitbook install
```

7. Проверете версията на GitBook с команда:

```
gitbook --version
```

Резултата може да изглежда по този начин:

```
CLI version: 2.3.2  
GitBook version: 3.2.3
```

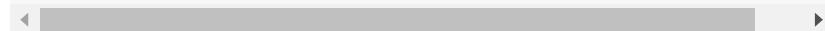
8. Стаптирайте процедурата за генериране на електронната книга в избран от Вас формат:

- **HTML**

```
gitbook build
```

Резултат от успешно изпълнение ще изглежда така:

```
info: 8 plugins are installed
info: 7 explicitly listed
info: loading plugin "hints"... OK
info: loading plugin "highlight"... OK
info: loading plugin "search"... OK
info: loading plugin "lunr"... OK
info: loading plugin "sharing"... OK
info: loading plugin "fontsettings"... OK
info: loading plugin "theme-default"... OK
info: found 35 pages
info: found 109 asset files
warn: "this.generator" property is deprecated, use "this.output.name" instead
warn: "navigation" property is deprecated
warn: "book" property is deprecated, use "this" directly instead
warn: "options" property is deprecated, use config.get(key) instead
info: >> generation finished with success in 2.2s !
```

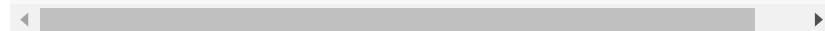


- **PDF**

```
gitbook pdf
```

Резултат от успешно изпълнение ще изглежда така:

```
info: 8 plugins are installed
info: 7 explicitly listed
info: loading plugin "hints"... OK
info: loading plugin "highlight"... OK
info: loading plugin "search"... OK
info: loading plugin "lunr"... OK
info: loading plugin "sharing"... OK
info: loading plugin "fontsettings"... OK
info: loading plugin "theme-default"... OK
info: found 35 pages
info: found 109 asset files
warn: "this.generator" property is deprecated, use "this.output.name" instead
warn: "navigation" property is deprecated
warn: "book" property is deprecated, use "this" directly instead
warn: "options" property is deprecated, use config.get(key) instead
info: >> generation finished with success in 18.1s !
info: >> 1 file(s) generated
```



- **EPUB**

```
gitbook epub
```

Резултат от успешно изпълнение ще изглежда така:

```
info: 8 plugins are installed
info: 7 explicitly listed
info: loading plugin "hints"... OK
info: loading plugin "highlight"... OK
info: loading plugin "search"... OK
info: loading plugin "lunr"... OK
info: loading plugin "sharing"... OK
info: loading plugin "fontsettings"... OK
info: loading plugin "theme-default"... OK
info: found 35 pages
info: found 110 asset files
warn: "this.generator" property is deprecated, use "this.output.name" instead
warn: "navigation" property is deprecated
warn: "book" property is deprecated, use "this" directly instead
warn: "options" property is deprecated, use config.get(key) instead
info: >> generation finished with success in 9.6s !
info: >> 1 file(s) generated
```

• MOBI

```
gitbook mobi
```

Резултат от успешно изпълнение ще изглежда така:

```
info: 8 plugins are installed
info: 7 explicitly listed
info: loading plugin "hints"... OK
info: loading plugin "highlight"... OK
info: loading plugin "search"... OK
info: loading plugin "lunr"... OK
info: loading plugin "sharing"... OK
info: loading plugin "fontsettings"... OK
info: loading plugin "theme-default"... OK
info: found 35 pages
info: found 111 asset files
warn: "this.generator" property is deprecated, use "this.output.name" instead
warn: "navigation" property is deprecated
warn: "book" property is deprecated, use "this" directly instead
warn: "options" property is deprecated, use config.get(key) instead
info: >> generation finished with success in 8.5s !
info: >> 1 file(s) generated
```

Linux

1. Актуализирайте вашата дистрибуция:

```
sudo apt update -y && sudo apt upgrade
```

2. Инсталирайте необходимите модули:

```
sudo apt install -y calibre  
sudo apt install -y nodejs  
sudo apt install -y npm  
sudo npm install -g ebook-convert  
sudo npm install -g graceful-fs@4.2.0  
sudo npm install -g gitbook-cli@2.1.2  
gitbook install
```

3. Клонирайте електронното хранилище на книгата:

```
git clone https://github.com/dimitarminchev/DCPA.git  
cd DCPA
```

4. Генерирайте електронните книги:

```
gitbook pdf  
gitbook mobi  
gitbook epub
```

Резултат от успешно изпълнение ще изглежда така:

```
info: 8 plugins are installed  
info: 7 explicitly listed  
info: loading plugin "hints"... OK  
info: loading plugin "highlight"... OK  
info: loading plugin "search"... OK  
info: loading plugin "lunr"... OK  
info: loading plugin "sharing"... OK  
info: loading plugin "fontsettings"... OK  
info: loading plugin "theme-default"... OK  
info: found 36 pages  
info: found 80 asset files  
warn: "this.generator" property is deprecated, use "this.output.name" instead  
warn: "navigation" property is deprecated  
warn: "book" property is deprecated, use "this" directly instead  
warn: "options" property is deprecated, use config.get(key) instead  
info: >> generation finished with success in 34.8s !  
info: >> 1 file(s) generated
```

Книга



Настоящата електронна книга "Developing Cross-Platform Apps" е структурирана в две глави. Глава 1 е озаглавена "Универсални Windows приложения" и въвежда читателя във вълнуващия свят на Microsoft технологиите за разработка на универсални приложения за платформа Windows. Глава 2 е озаглавена "Мултиплатформени мобилни приложения" и въвежда читателя във вълнуващия свят на Microsoft технологиите за разработка на мултиплатформени мобилни приложения.

Автор

Димитър Минчев е университетски преподавател към Център по информатика и технически науки при Бургаски свободен университет. Създава уникалните [Академията за таланти по програмиране](#) и [Школа по роботика](#) за ученици от Бургас. Подготвя студенти за [Републиканска студенческа олимпиада по програмиране](#) в [Клуб по състезателно програмиране](#). Организира съревнование за разработка на настолни и мобилни приложения на морето [ХАКАТОН @ БСУ](#). Инициира ученическото състезание по програмиране [CODE@BURGAS](#). Преподавател от националната програма [Обучение за ИТ кариера](#) на Министерството на образованието и науката.

- **Служебен:** +359 56 900 477 и mitko@bfu.bg
- **Личен:** +359 899 148 872 и dimitar.minchev@gmail.com
- **Блог:** <http://www.minchev.eu>

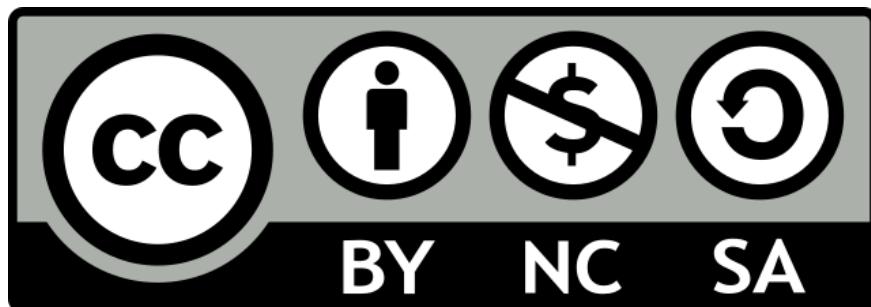
Съвет

Пожелавам Ви много забавление, провали и успехи използваки Microsoft технологиите за разработка на универсални Windows приложения и мултиплатформени мобилни приложения.

Лиценз

Предупреждение

Всиччи материали от настоящата електронна книга се разпространяват под лиценз **CC-BY-NC-SA**.



Имате право на:

Споделяте — да копирате и повторно да разпространявате материала на всякахъв носител или във всякахъв формат

Адаптирате — да преработвате, преобразувате и доразвивате материала
Ако спазвате условията на лиценза, лицензодателят не може да отмени тези
свободи.

При следните условия:

Признание — Вие сте длъжни да посочите автора, да дадете електронна
препратка към лиценза, и да укажете дали сте внесли промени. Можете да
 направите това по всеки разумен начин, но не по начин, който предполага,
 че лицензодателят одобрява Вас или използването от Ваша страна на
 материала. Ако спазвате условията на лиценза, лицензодателят не може да
 отмени тези свободи.

Некомерсиално — Вие нямаете право да използвате материала за търговски
 цели.

Споделяне на споделеното — ако преработвате, преобразувате или
 доразвивате материала, Вие сте длъжни да разпространявате своите
 приноси съгласно същия лиценз като оригиналa.

Без допълнителни ограничения — Вие нямаете право да прилагате правни
 условия или технологични мерки които създават правни ограничения за
 други лица да извършват разрешеното от лиценза.

Информация

Пълният текст на лиценза е достъпен в Интернет на адрес: CC-BY-NC-SA

Формат

1. Учебното пособие е налично за свободно четене под формата на GitBook базиран [електронен формат](#).
2. Учебните ресурси са налични за свободно изтегляне от GitHub базирано [електронно хранилище](#).

Формат	ISBN	Barcode
PDF	978-619-7126-66-2	 9 786197 126662
MOBI	978-619-7126-67-9	 9 786197 126679
EPUB	978-619-7126-68-6	 9 786197 126686