

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
Clicker Mania 3.0	1.3.2.4
HTML Downloader 2.0	1.3.2.5
RSS Reader 2.0	1.3.2.6
JSON Reader 2.0	1.3.2.7
Заключение	1.4

Въведение

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.

Формат и лиценз

Учебното пособие е налично за свободно четене под формата на GitBook базиран [електронен формат](#). Учебните ресурси са налични за свободно изтегляне от GitHub базирано [електронно хранилище](#). Всични учебни материали се разпространяват под безплатен 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. Създава уникалните [Академията за таланти по програмиране](#) и [Школа по роботика](#) за ученици от Бургас. Подготвя студенти за [Републиканска студента 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.

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

Универсални Windows приложения

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

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 Алън през 1975 година.



Фиг. 1. Пол Алън (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.

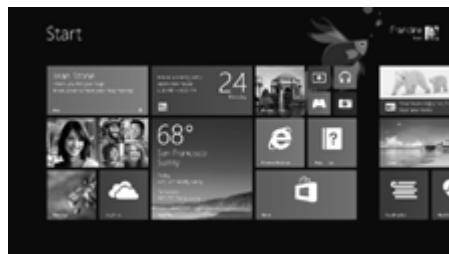


Фиг. 2. Microsoft Windows 1.0 излиза през 1983

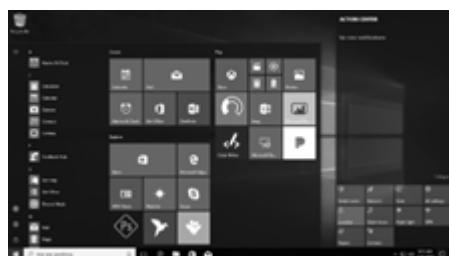


Фиг. 3. Microsoft Word 1.0 излиза през 1989

Операционните системи Windows 8 и 8.1 са представени съответно на 1 Август 2012 и 27 Август 2013. Подръжката 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



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

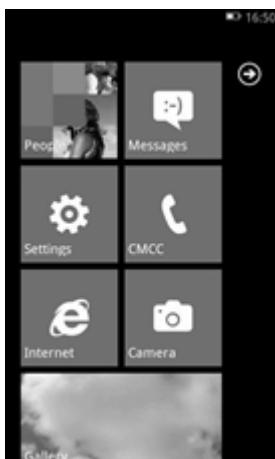
Табл. 2 Публични версии на операционната система Windows 10

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 постоянната свързаност към Интернет, която е добре комбинирана с мощн хардуер, осигуряващ оптимален комфорт при ползване с разнообразна мултимедия.



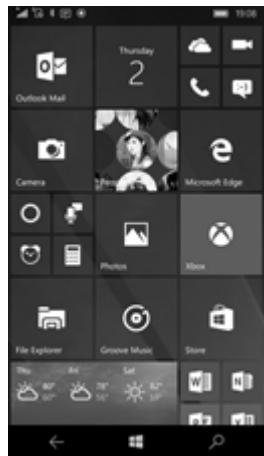
Фиг.6. Windows Phone 7 излиза през 2010

Първият Windows Phone 7 е представен на 21 октомври 2010, локализиран е на 25 езика, достърен в 35 страни, а приложението 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. Приложението за него се публикуват в 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

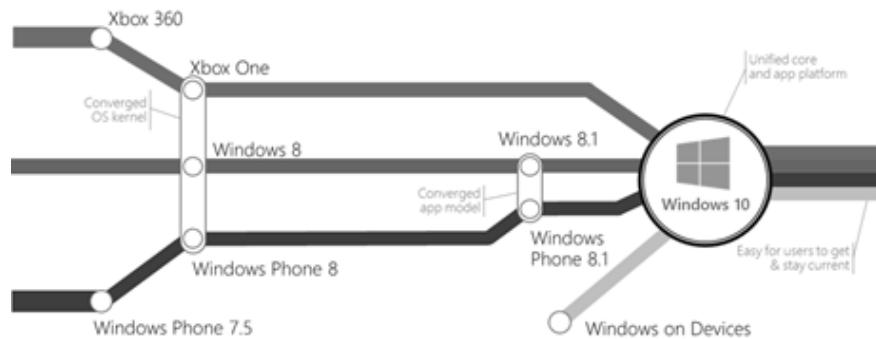
На 20 ноември 2015 е представена операционната система Windows 10

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.

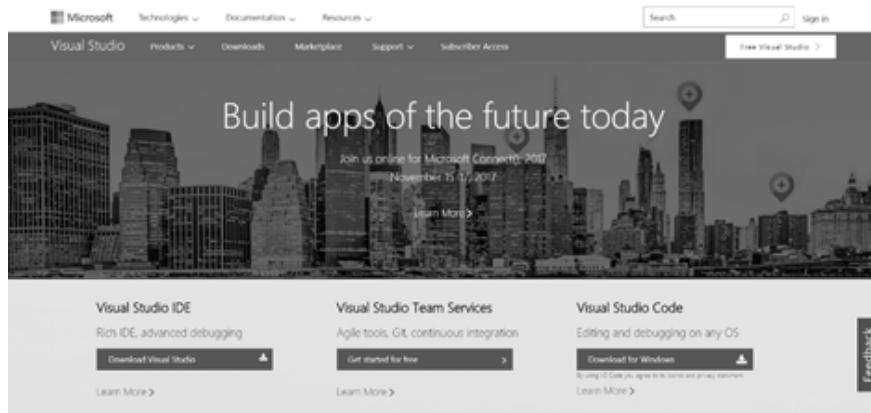


Фиг.10. Архитектура на универсалната Windows платформа за приложения

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

Инсталация

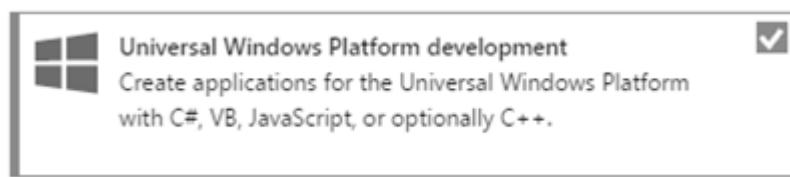
Напълно бесплатна версия на интегрираната среда за разработка наречена **Microsoft Visual Studio Community Edition**, може да се изтегли от официалният сайт на компанията в Интернет на адрес: <https://www.visualstudio.com/>. Понастоящем последната актуална версия на този продукт е Microsoft Visual Studio Community Edition 2017.



Фиг. 11. Microsoft Visual Studio Website

Стандартна инсталация по подразбиране изиска скромните **594 MB** 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 приложения за платформа Windows е необходимо да се включи опцията **Universal Windows Platform development** (Фиг. 12), като се добавят допълнителни софтуерни компоненти за разработчици (Фиг. 13).



Фиг. 12. Visual Studio 2017 Installer: Universal Windows Platform development

- ✓ Universal Windows Platform development
 - Included
 - ✓ Blend for Visual Studio
 - ✓ .NET Native
 - ✓ NuGet package manager
 - ✓ Universal Windows Platform tools
 - ✓ Windows 10 SDK (10.0.16299.0) for UWP: C#, VB...

Optional

- Windows 10 SDK (10.0.15063.0) for UWP
- C++ Universal Windows Platform tools
- Graphics debugger and GPU profiler for DirectX
- Windows 10 SDK (10.0.14393.0)
- Windows 10 SDK (10.0.10586.0)
- Windows 10 SDK (10.0.10240.0)

Фиг. 13. Visual Studio 2017 Installer: UWP Windows 10 SDK

Допълнителните инструменти за разработчици увеличават обема на необходимото дисково пространство за инсталациране на интегрираната среда за разработка Visual Studio. Необходими са **39 GB** свободно дисково пространство (Фиг. 14).



Фиг. 14. Visual Studio 2017 Installer

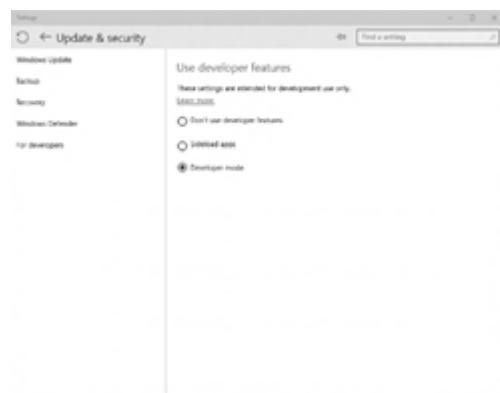
Продължава се напред и се изчаква приключването на инсталацията. Процедурата отнема значително количество време.

Включване на режима за разработчици

За да разработвате универсални приложения за платформа Windows е необходимо да включите операционната система в режим за разработчици, както е показано на Фиг. 15 и Фиг. 16.



Фиг. 15. Включване на режима за разработчици на мобилно устройство



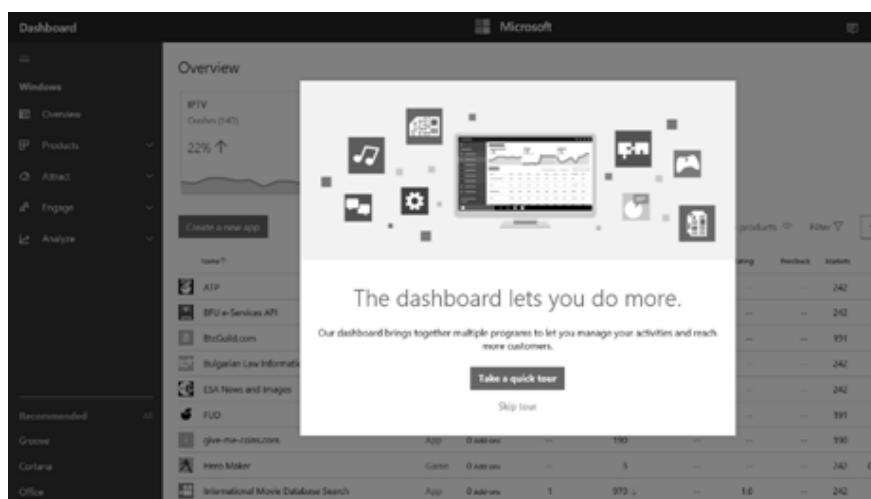
Фиг. 16 Включване на режима за разработчици на персонален компютър

Регистриране като разработчик на приложения

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

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

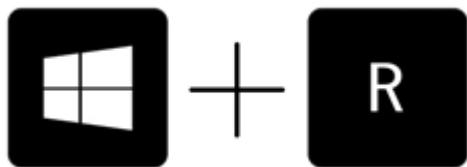
Регистрацията като разработчик на приложения в центъра за разработчици се заплаща с еднократна такса от около 19 американски долара за индивидуални потребители и около 99 американски долара за корпоративни клиенти. Студенти и преподаватели могат да се регистрират бесплатно, ако имат абонамент и използват програмата **Microsoft Imagine**.



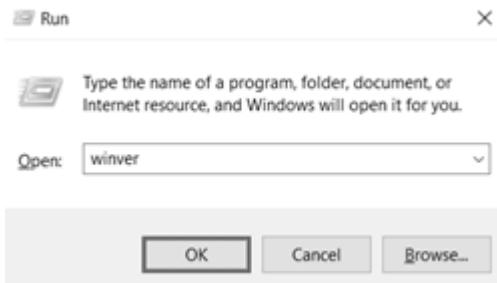
Фиг. 17. Център за разработчици на Microsoft

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 версия на операционната система.



Фиг.18. Клавишина комбинация Windows Key + R



Фиг.19. Диалогов прозорец RUN

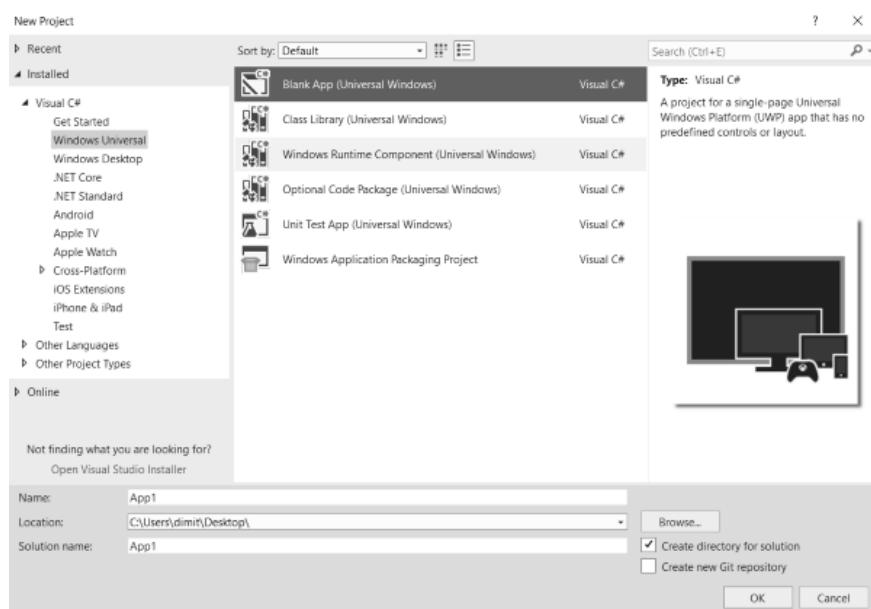


Фиг.20. Версия на операционната система Windows

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. Създаване на нов проект в интегрираната среда за разработка Visual Studio

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

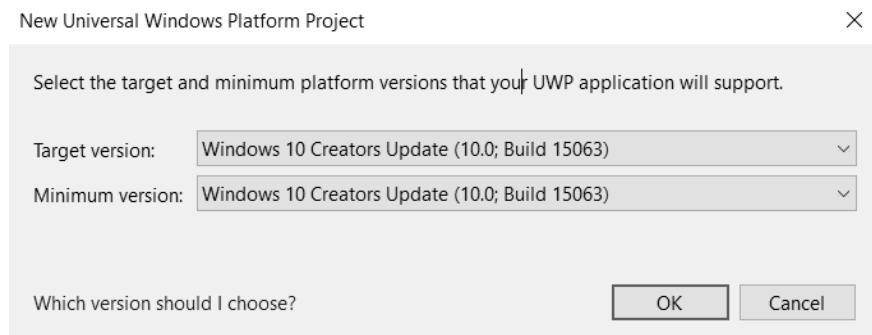
- Microsoft Windows 10 Version 1809 Build 10.0.17763.134
- Microsoft Visual Studio Community 2017 Version 15.9.2

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 появилния 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.

```

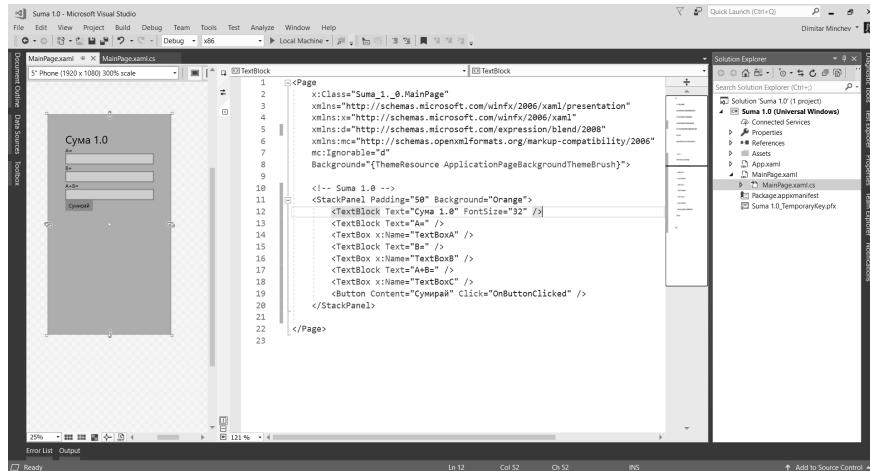
<Page
    x:Class="Suma_1._0.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"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

    <!-- User Interface (UI): Suma 1.0 -->
    <StackPanel Padding="50" Background="Orange">
        <TextBlock Text="Сума 1.0" FontSize="32" />
        <TextBlock Text="A=" />
        <TextBox x:Name="TextBoxA" />
        <TextBlock Text="B=" />
        <TextBox x:Name="TextBoxB" />
        <TextBlock Text="A+B=" />
        <TextBox x:Name="TextBoxC" />
        <Button Content="Сумирай" 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.

```

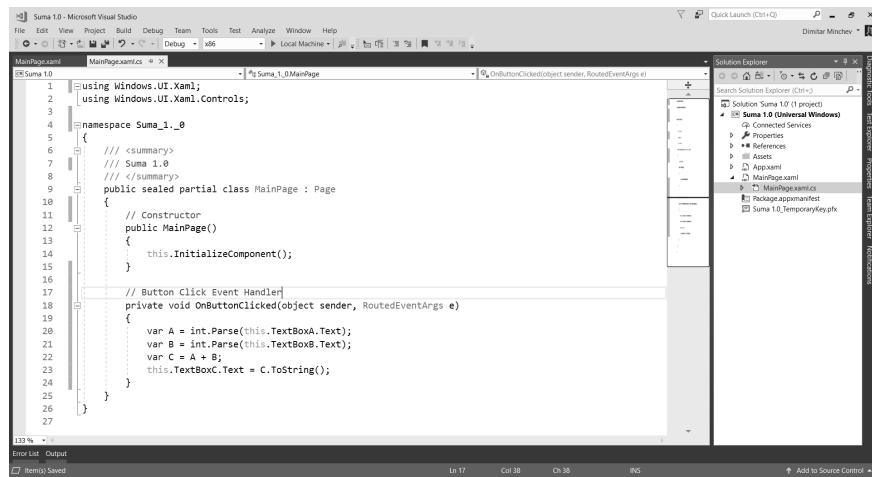
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;

namespace Suma_1._0
{
    // Business Logic (BLS): Suma 1.0
    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.TextBoxA.Text);
            var B = int.Parse(this.TextBoxB.Text);
            var C = A + B;
            this.TextBoxC.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 клавиши **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.

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 появилия 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.

```

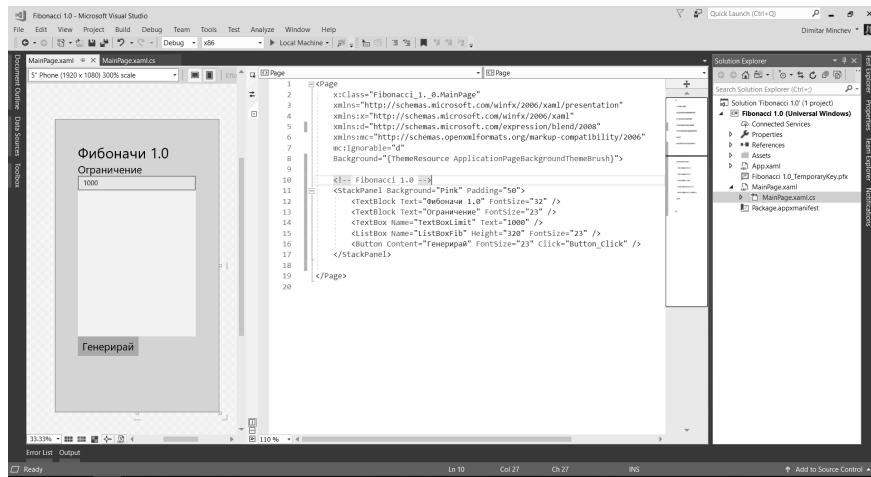
<Page
    x:Class="Fibonacci_1_0.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"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

    <!-- User Interface (UI): Fibonacci 1.0 -->
    <StackPanel Background="Pink" Padding="50">
        <TextBlock Text="Фибонacci 1.0" FontSize="32" />
        <TextBlock Text="Ограничение" FontSize="23" />
        <TextBox Name="TextBoxLimit" Text="1000" />
        <ListBox Name="ListBoxFib" Height="320" FontSize="23" />
        <Button Content="Генерирай" FontSize="23" 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. 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 System.Collections.ObjectModel;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;

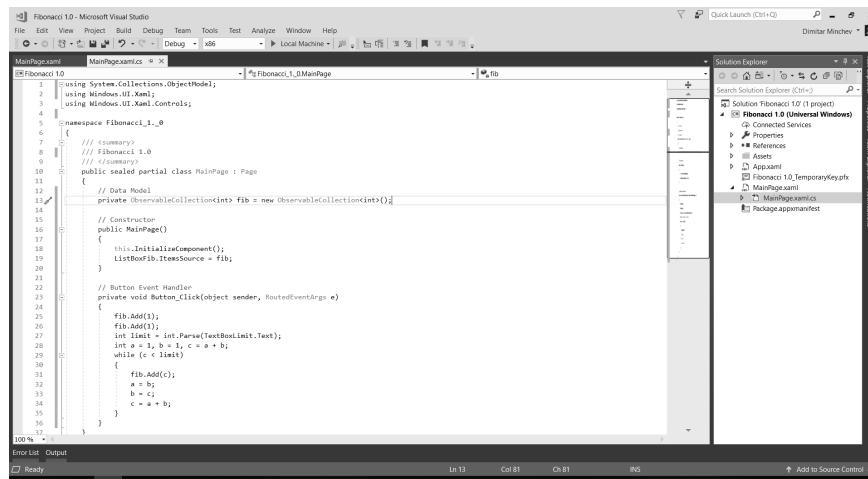
namespace Fibonacci_1._0
{
    // Business Logic (BL): Fibonacci 1.0
    public sealed partial class MainPage : Page
    {
        // Collection
        private ObservableCollection<int> fib = new ObservableCollection<int>();

        // Constructor
        public MainPage()
        {
            this.InitializeComponent();
            ListBoxFib.ItemsSource = fib;
        }

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

```

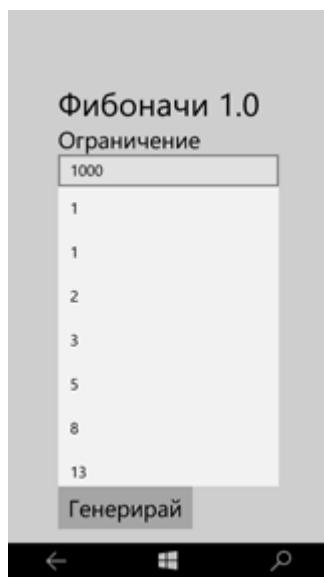
Изглед от бизнес логиката (C#) в интегрираната среда за разработка Visual Studio по време на разработване на приложението:



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.

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**.

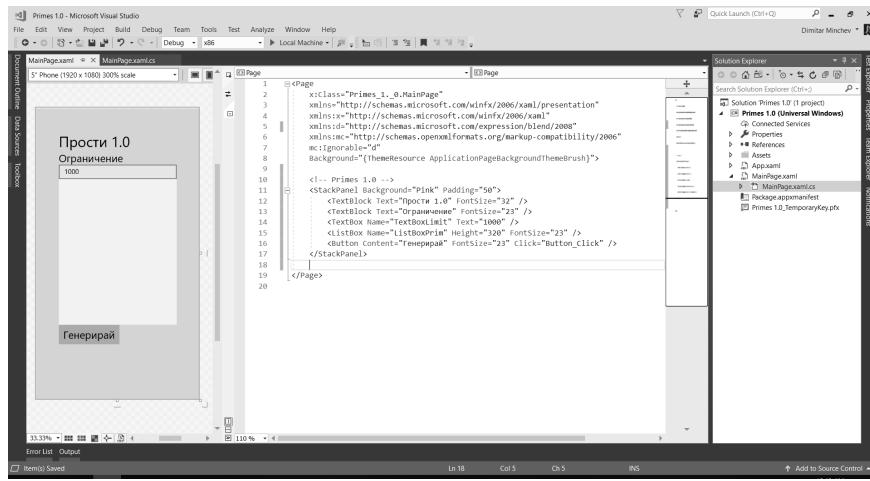
MainPage.xaml

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: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="Pink" Padding="50">  
        <TextBlock Text="Прости 1.0" FontSize="32" />  
        <TextBlock Text="Ограничение" FontSize="23" />  
        <TextBox Name="TextBoxLimit" Text="1000" />  
        <ListBox Name="ListBoxPrim" Height="320" FontSize="23" />  
        <Button Content="Генерирай" FontSize="23" 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. 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(TextBoxLimit.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) ListBoxPrim.Items.Add(k);
            }
        }
    }
}

```

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

The screenshot shows the Microsoft Visual Studio interface. The code editor displays the `MainPage.xaml.cs` file with C# code for generating prime numbers. The Solution Explorer on the right shows the project structure for "Primes 1.0" with files like `Primes_1_0.csproj`, `Properties`, `Assets`, `MainPage.xaml`, and `MainPage.xaml.cs`.

```

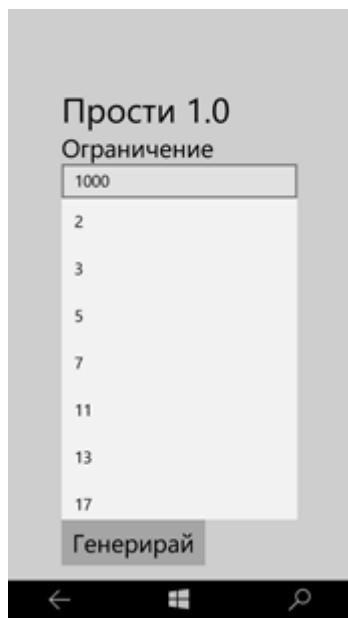
1 //using Windows.UI.Xaml;
2 //using Windows.UI.Xaml.Controls;
3
4 namespace Primes_1_0
5 {
6     /// <summary>
7     /// Primes 1.0
8     /// </summary>
9     public sealed partial class MainPage : Page
10    {
11        // Constructor
12        public MainPage()
13        {
14            this.InitializeComponent();
15        }
16
17        // Button Event Handler
18        private void button_Click(object sender, RoutedEventArgs e)
19        {
20            int limit = int.Parse(textBoxLimit.Text);
21            for (int k = 2; k < limit; k++)
22            {
23                bool prime = true;
24                for (int j = 2; j < k; j++)
25                    if (k % j == 0) prime = false;
26                if (prime) listBoxPrim.Items.Add(k);
27            }
28        }
29    }
30

```

Фиг. 30. Изглед от бизнес логиката на разработваното приложение

Demo

Стартирайте приложението от менюто: **Debug > Start Debugging** или като натиснете клавиш **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
    {
        // Phone Book 1.0
        public static ObservableCollection<Contact> people = 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: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="Pink" Padding="20">
        <TextBlock Text="Add Contact" FontSize="42" />
        <TextBlock Text="picture" FontSize="24" />
        <TextBox Name="picture" />
        <TextBlock Text="name" FontSize="24" />
        <TextBox Name="name" />
        <TextBlock Text="phone" FontSize="24" />
        <TextBox Name="phone" />
        <Button Content="Add" Click="Button_Click" />
    </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 contact = new Contact(new Uri(picture.Text), name.Text, phone.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: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="Pink" Padding="20">
        <TextBlock Text="PhoneBook 1.0" FontSize="42" />
        <ListBox Name="ListBoxPeople" Height="320">
            <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>
        <Button Content="Add Contact" Click="Button_Click" />
    </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();
            ListBoxPeople.ItemsSource = App.people;
        }

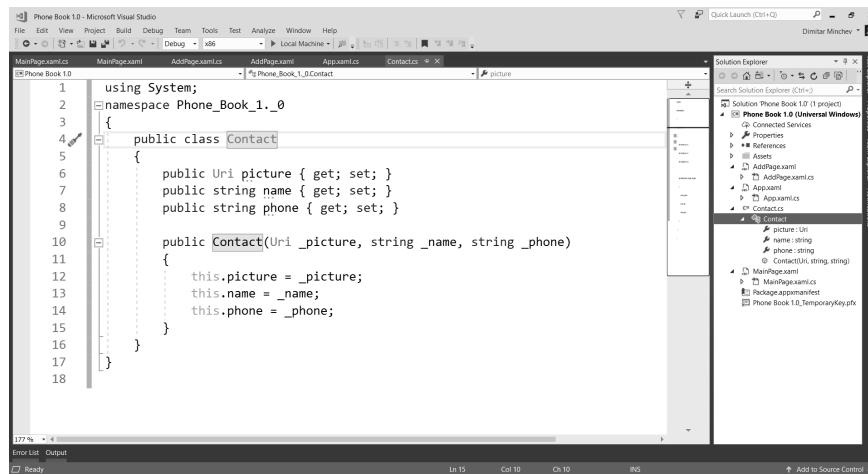
        // Button Click Event Handler
        private void Button_Click(object sender, RoutedEventArgs e)
        {
            this.Frame.Navigate(typeof(AddPage));
        }

        // Navigate
        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            if (e.Parameter is Contact)
                App.people.Add(e.Parameter as Contact);
        }
    }
}

```

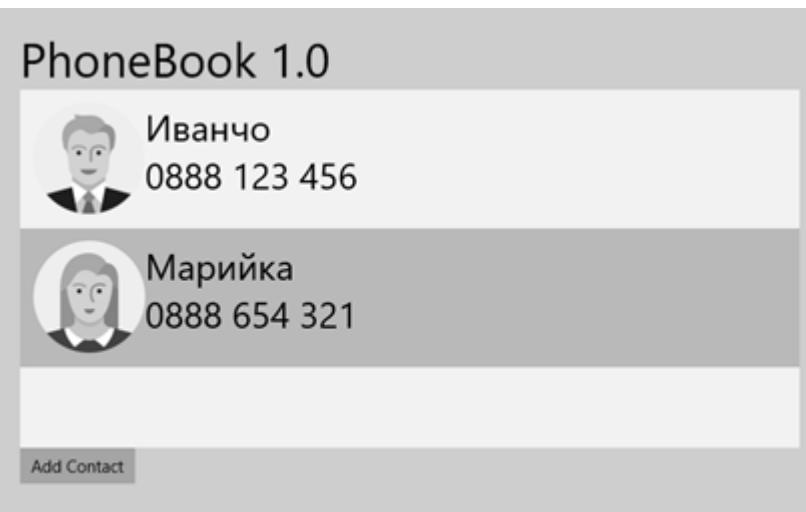
Demo

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 клавиши **F5**.



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

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. Стартирайте 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.

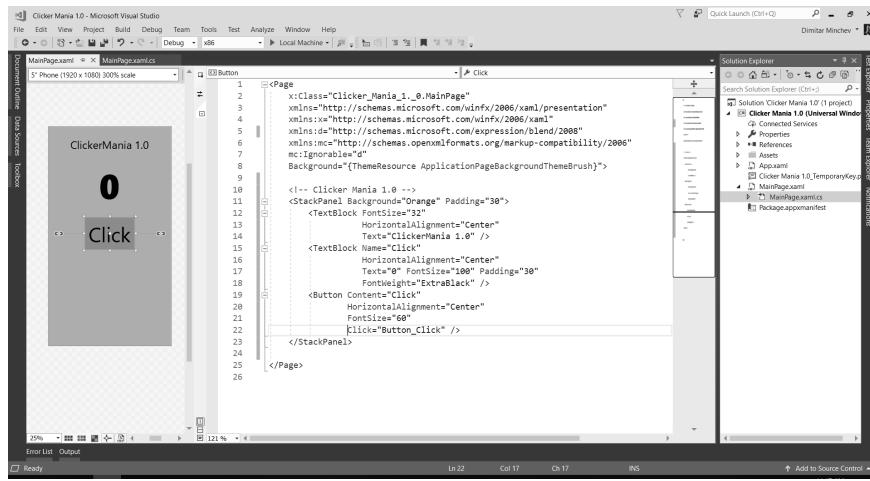
Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

```
<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: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="30">
        <TextBlock FontSize="32" HorizontalAlignment="Center" Text="ClickerMania 1.0" />
        <TextBlock Name="Click" HorizontalAlignment="Center" Text="0" FontSize="100" />
        <Button Content="Click" HorizontalAlignment="Center" FontSize="60" Click="Button_Click" />
    </StackPanel>

</Page>
```

Изглед от дизайна на потребителския 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 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
    {
        // var
        private int counter = 0;
        private StorageFile file;
        private StorageFolder local = ApplicationData.Current.LocalFolder;

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

        // Button Click Event Handler
        private async void Button_Click(object sender, RoutedEventArgs e)
        {
            counter++;
            Click.Text = counter.ToString();
            // Save to file
            byte[] bytes = System.Text.Encoding.UTF8.GetBytes(Click.Text.ToCharArray());
            var _file = await local.CreateFileAsync("clicker.txt", CreationCollisionOption.ReplaceExisting);
            using (var _stream = await _file.OpenStreamForWriteAsync()) _stream.Write(
        }

        // Read from File
        private async void Read()
        {
            try
            {
                file = await local.GetFileAsync("clicker.txt");
                if (file == null) file = await local.CreateFileAsync("clicker.txt");
                else
                {
                    Stream stream = await file.OpenStreamForReadAsync();
                    StreamReader reader = new StreamReader(stream);
                    Click.Text = reader.ReadToEnd();
                    if (Click.Text == "")
                    {
                        Click.Text = "0";
                        counter = 0;
                    }
                    else counter = int.Parse(Click.Text);
                }
            }
            catch { ; }
        }
    }
}

```

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

```

26 // Button Even Handler
27 private async void Button_Click(object sender, RoutedEventArgs e)
28 {
29     counter++;
30     Click.Text = counter.ToString();
31     // Save to file
32     byte[] bytes = System.Text.Encoding.UTF8.GetBytes(Click.Text.ToCharArray());
33     var _file = await local.CreateFileAsync("clicker.txt", CreationCollisionOption.ReplaceExisting);
34     using (var _stream = await _file.OpenStreamForWriteAsync()) _stream.Write(bytes, 0, bytes.Length);
35 }
36
37 // Read from File
38 private async void Read()
39 {
40     try
41     {
42         file = await local.GetFileAsync("clicker.txt");
43         if (file == null) file = await local.CreateFileAsync("clicker.txt");
44         else
45         {
46             Stream stream = await file.OpenStreamForReadAsync();
47             StreamReader reader = new StreamReader(stream);
48             Click.Text = reader.ReadToEnd();
49             if (Click.Text == "")
50             {
51                 Click.Text = "0";
52                 counter = 0;
53             }
54             else counter = int.Parse(Click.Text);
55         }
56     }
57 }

```

Фиг. 35. Изглед от бизнес логиката на разработваното приложение

Demo

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

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

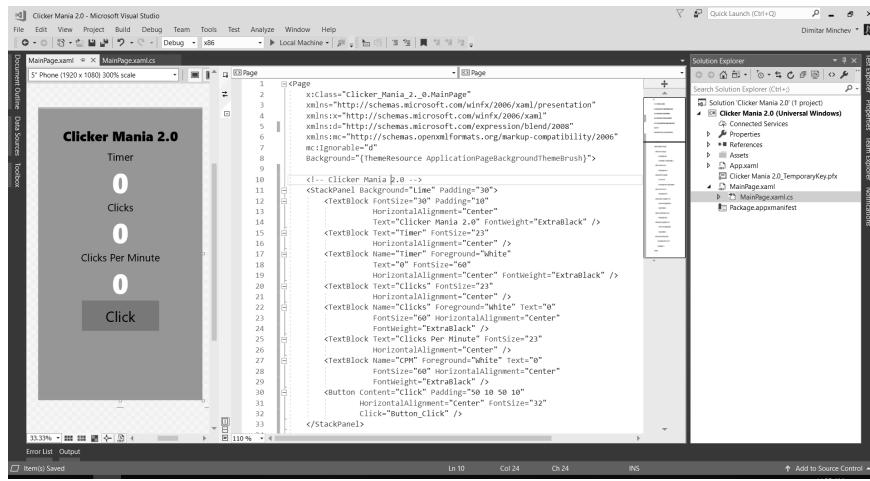
Копирайте (Ctrl+C) и поставете (Ctrl+V) програмният фрагмент даден по-долу във Вашето приложение.

```
<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: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">
        <TextBlock FontSize="30" Padding="10" HorizontalAlignment="Center" Text="Click" />
        <TextBlock Text="Timer" FontSize="23" HorizontalAlignment="Center" />
        <TextBlock Name="Timer" Foreground="White" Text="0" FontSize="60" HorizontalAlignment="Center" />
        <TextBlock Text="Clicks" FontSize="23" HorizontalAlignment="Center" />
        <TextBlock Name="Clicks" Foreground="White" Text="0" FontSize="60" HorizontalAlignment="Center" />
        <TextBlock Text="Clicks Per Minute" FontSize="23" HorizontalAlignment="Center" />
        <TextBlock Name="CPM" Foreground="White" Text="0" FontSize="60" HorizontalAlignment="Center" />
        <Button Content="Click" Padding="50 10 50 10" HorizontalAlignment="Center" For
    </StackPanel>

</Page>
```

Изглед от дизайна на потребителския 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();

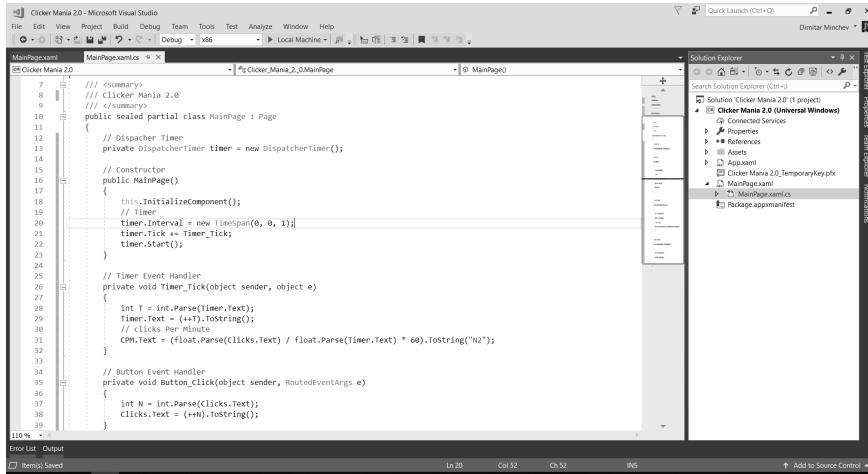
        // Constructor
        public MainPage()
        {
            this.InitializeComponent();
            // Timer
            timer.Interval = new TimeSpan(0, 0, 1);
            timer.Tick += Timer_Tick;
            timer.Start();
        }

        // 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).ToString();
        }

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

```

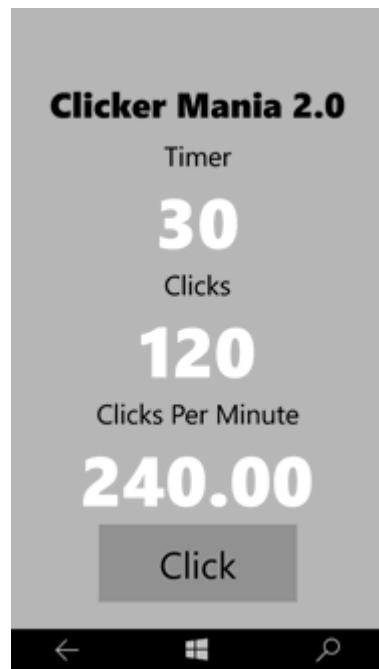
Изглед от бизнес логиката (C#) в интегрираната среда за разработка Visual Studio по време на разработване на приложението:



Фиг. 38. Изглед от бизнес логиката на разработваното приложение

Demo

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

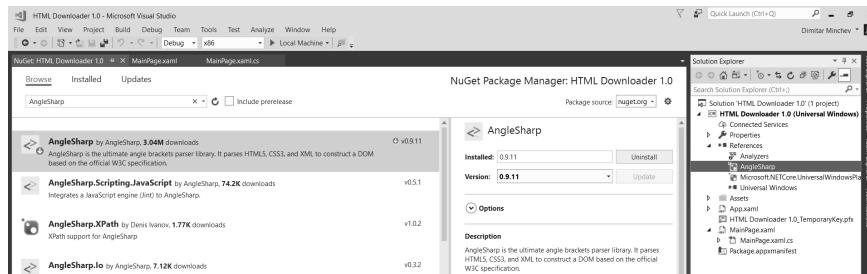
HTML е основният маркиращ език за описание и дизайн на уеб страници. HTML е стандарт в Интернет, а правилата се определят от международния консорциум W3C.

- Източник: [Wikipedia](#)

Start

1. Стартирайте 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 фигуранта:



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 0.9.11
```

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.

```

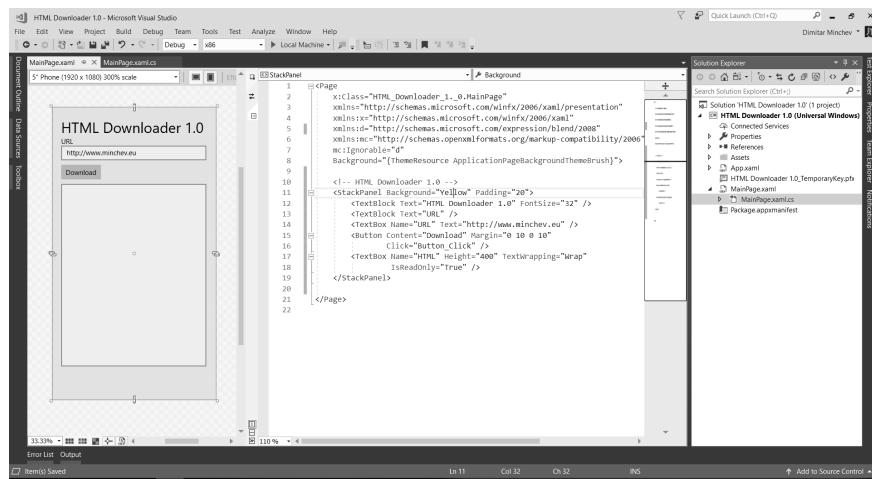
<Page
    x:Class="HTML_Downloader_1._0.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"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

    <!-- User Interface (UI): HTML Downloader 1.0 -->
    <StackPanel Background="Yellow" Padding="20">
        <TextBlock Text="HTML Downloader 1.0" FontSize="32" />
        <TextBlock Text="URL" />
        <TextBox Name="URL" Text="http://www.minchev.eu" />
        <Button Content="Download" Margin="0 10 0 10" Click="Button_Click" />
        <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 AngleSharp.Parser.Html;
using System;
using System.Net.Http;
using System.Threading.Tasks;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;

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();
        }

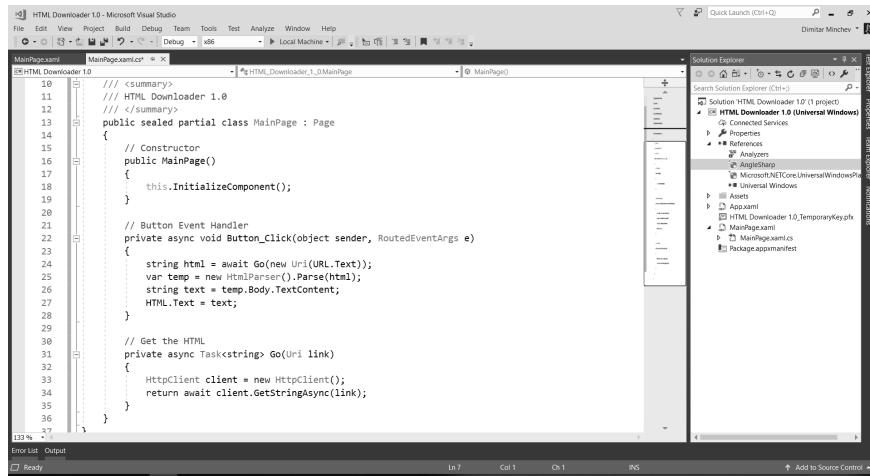
        // Button Click Event Handler
        private async void Button_Click(object sender, RoutedEventArgs e)
        {
            string html = await Go(new Uri(URL.Text));
            var temp = new HtmlParser().Parse(html);
            string text = temp.Body.TextContent;
            HTML.Text = text;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

        // Get the HTML
        private async Task<string> Go(Uri link)
        {
            HttpClient client = new HttpClient();
            return await client.GetStringAsync(link);
        }
    }
}

```

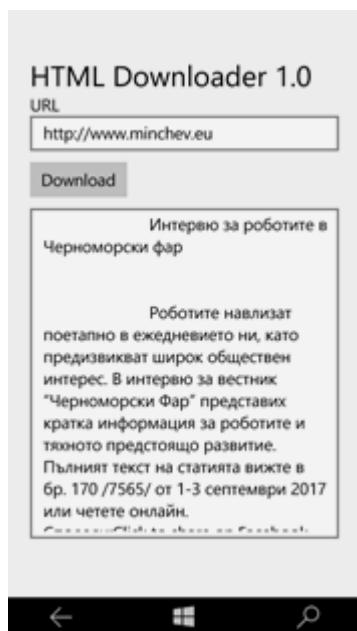
Demo

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



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.

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**.

FeedItem.cs

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

```
namespace RSS_Reader_1._0
{
    // RSS Feed Item
    public class FeedItem
    {
        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.

```

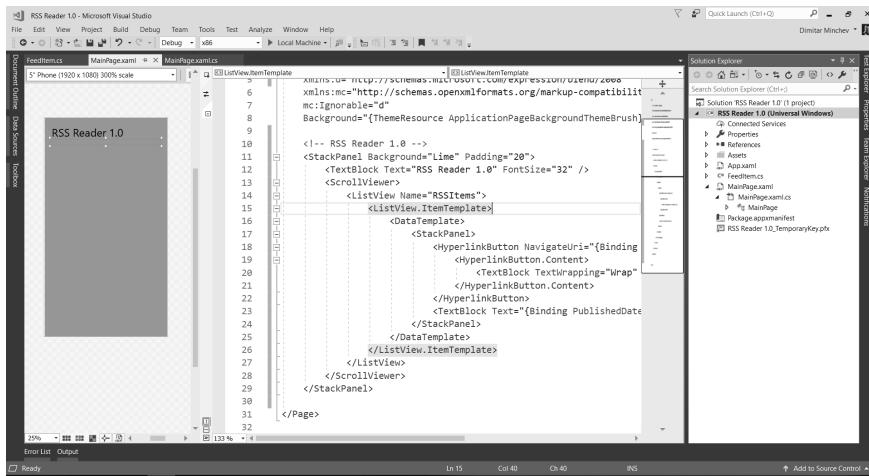
<Page
    x:Class="RSS_Reader_1._0.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"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

    <!-- User Interface (UI): RSS Reader 1.0 -->
    <StackPanel Background="Lime" Padding="20">
        <TextBlock Text="RSS Reader 1.0" FontSize="32" />
        <ScrollViewer>
            <ListView Name="RSSItems">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <StackPanel>
                            <HyperlinkButton NavigateUri="{Binding Link}">
                                <HyperlinkButton.Content>
                                    <TextBlock TextWrapping="Wrap" Text="{Binding Title}" />
                                </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.Controls;
using Windows.Web.Syndication;

namespace RSS_Reader_1_0
{
    // Business Logic(BL): RSS Reader 1.0
    public sealed partial class MainPage : Page
    {
        // View Model
        private ObservableCollection<FeedItem> RSSFeed = new ObservableCollection<FeedItem>();

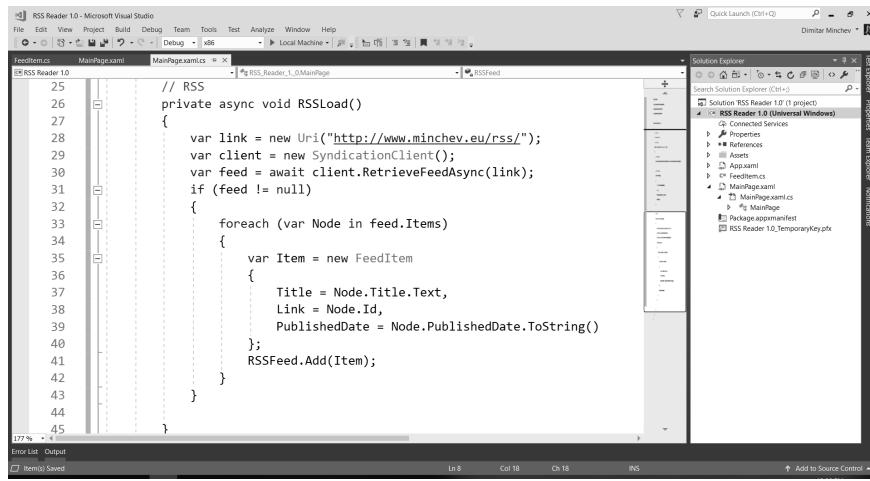
        // Constructor
        public MainPage()
        {
            this.InitializeComponent();

            // RSS
            RSSItems.ItemsSource = RSSFeed;
            RSSLoad();
        }

        // RSS
        private async void RSSLoad()
        {
            var link = new Uri("http://www.minchev.eu/rss/");
            var client = new SyndicationClient();
            var feed = await client.RetrieveFeedAsync(link);
            if (feed != null)
            {
                foreach (var Node in feed.Items)
                {
                    var Item = new FeedItem
                    {
                        Title = Node.Title.Text,
                        Link = Node.Id,
                        PublishedDate = Node.PublishedDate.ToString()
                    };
                    RSSFeed.Add(Item);
                }
            }
        }
    }
}
```

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**.

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 11.0.2
PM> Install-Package AngleSharp -Version 0.9.11
```

RootObject.cs

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

```
namespace JSON_Reader_1._0
{
    // https://api.chucknorris.io/jokes/random
    // http://json2csharp.com/
    public class RootObject
    {
        public object category { get; set; }
        public string icon_url { get; set; }
        public string id { get; set; }
        public string url { get; set; }
        public string value { 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.

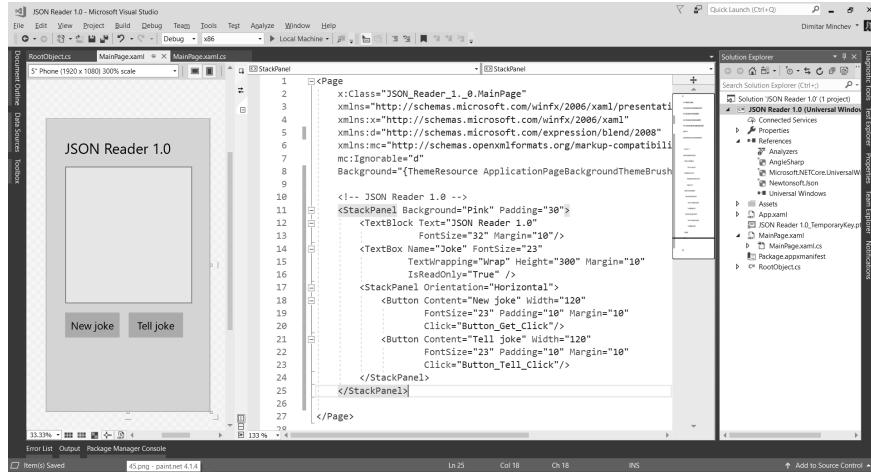
```

<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: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="Pink" Padding="30">
        <TextBlock Text="JSON Reader 1.0" FontSize="32" Margin="10"/>
        <TextBox Name="Joke" FontSize="23" TextWrapping="Wrap" Height="300" Margin="10" />
        <StackPanel Orientation="Horizontal">
            <Button Content="New joke" Width="120" FontSize="23" Padding="10" Margin="10" Click="Button_Get_Click"/>
            <Button Content="Tell joke" Width="120" FontSize="23" Padding="10" Margin="10" Click="Button_Tell_Click"/>
        </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.Media.SpeechSynthesis;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Newtonsoft.Json;
using AngleSharp.Parser.Html;

namespace JSON_Reader_1._0
{
    // Business Logic (BL): JSON Reader 1.0
    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"));

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

            // Parse the HTML
            var html = new HtmlParser().Parse(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();

                // The object for controlling the speech synthesis engine (voice).
                var synth = new SpeechSynthesizer();

                // Generate the audio stream from plain text.
                var stream = await synth.SynthesizeTextToStreamAsync(Joke.Text);

                // Send the stream to the media object.
                mediaElement.SetSource(stream, stream.ContentType);
                mediaElement.Play();
            }
        }
    }
}

```

Demo

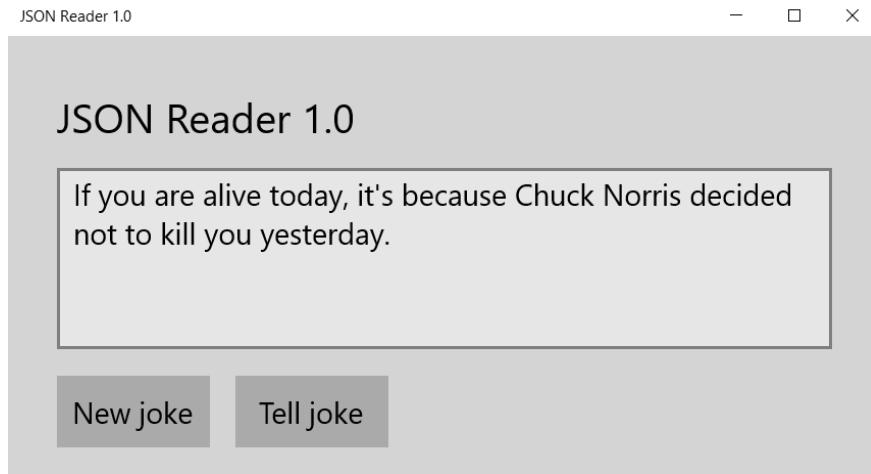
Изглед от бизнес логиката (C#) в интегрираната среда за разработка Visual Studio по време на разработване на приложението:

The screenshot shows the Microsoft Visual Studio interface with the following details:

- Title Bar:** JSON Reader 1.0 - Microsoft Visual Studio
- Menu Bar:** File, Edit, View, Project, Build, Debug, Item, Tools, Test, Analyze, Window, Help
- Toolbars:** Standard, Debug
- Code Editor:** RootObject.cs (MainPage.xaml.cs) is open. The code implements two button click handlers: `Button_Get_Click` and `Button_Tell_Click`. It uses HttpClient to download JSON from `https://api.chucknorris.io/jokes/random`, deserializes it into a `RootObject` class, parses the HTML content, and updates a `Joke` object with the text.
- Solution Explorer:** Shows the project structure for 'JSON Reader 1.0 (Universal Windows)' containing files like App.xaml, MainPage.xaml, MainPage.xaml.cs, Package.appxmanifest, and RootObject.cs.
- Status Bar:** Rem(n) Saved, In 33 Col 56 Ch 56 INS

Фиг. 48. Изглед от бизнес логиката на разработваното приложение

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



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

Мултиплатформени мобилни приложения

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

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 miliona разработчици от 120 страни използват Xamarin.

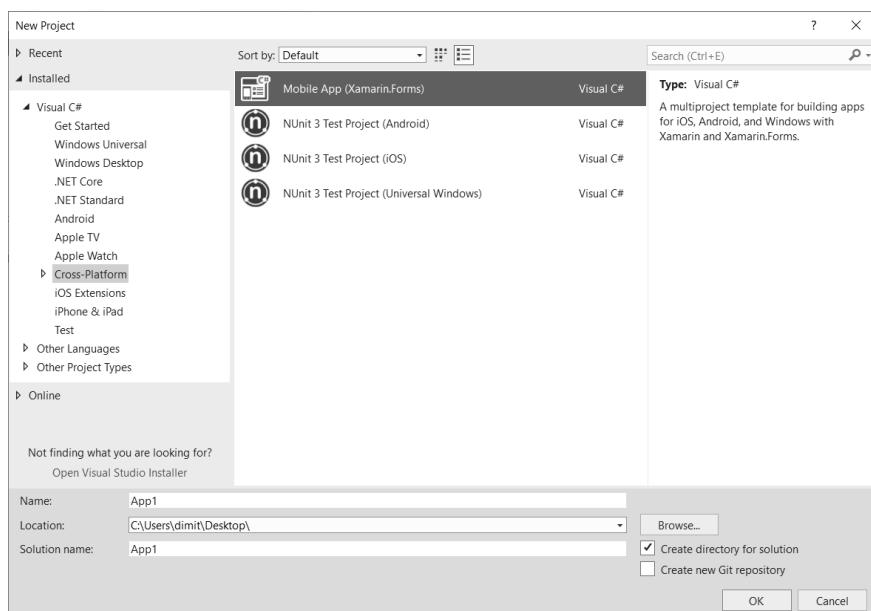


Фиг. 51 Платформено зависим и споделен изходен код

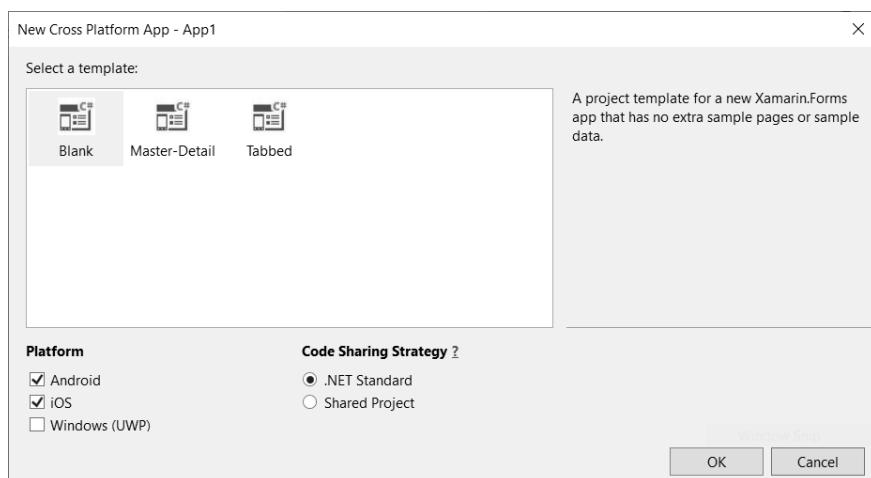
Приложения

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 мултиплатформен проект



Фиг.53 Конфигуриране на новия мултиплатформен проект

Изисквания

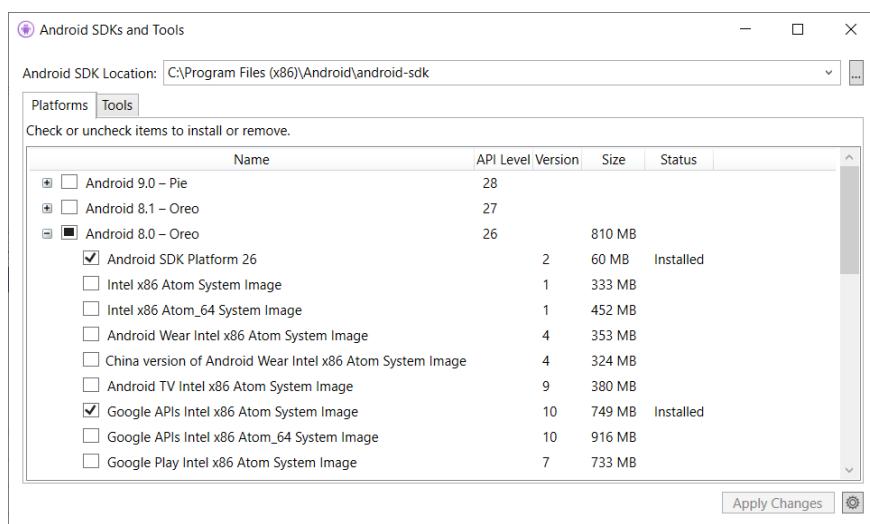
Примерите в тази част са разработени използвайки следните версии на операционната система и интегрираната среда за разработка:

- Microsoft Windows 10 Version 1809 Build 10.0.17763.134
- Microsoft Visual Studio Community 2017 Version 15.9.2

Настройки

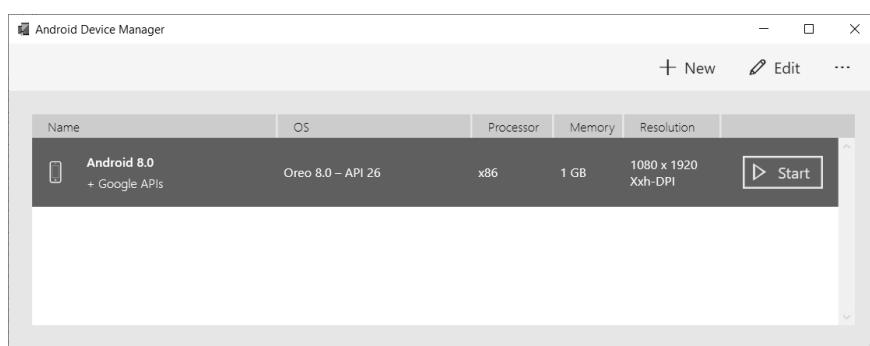
Меню **Tools > Android > SDK Manager** изберете и включете следните настройки:

- Android 8.0 - Oreo
- Android SDK Platform 26
- Google APIs Intel x86 Atom System Image



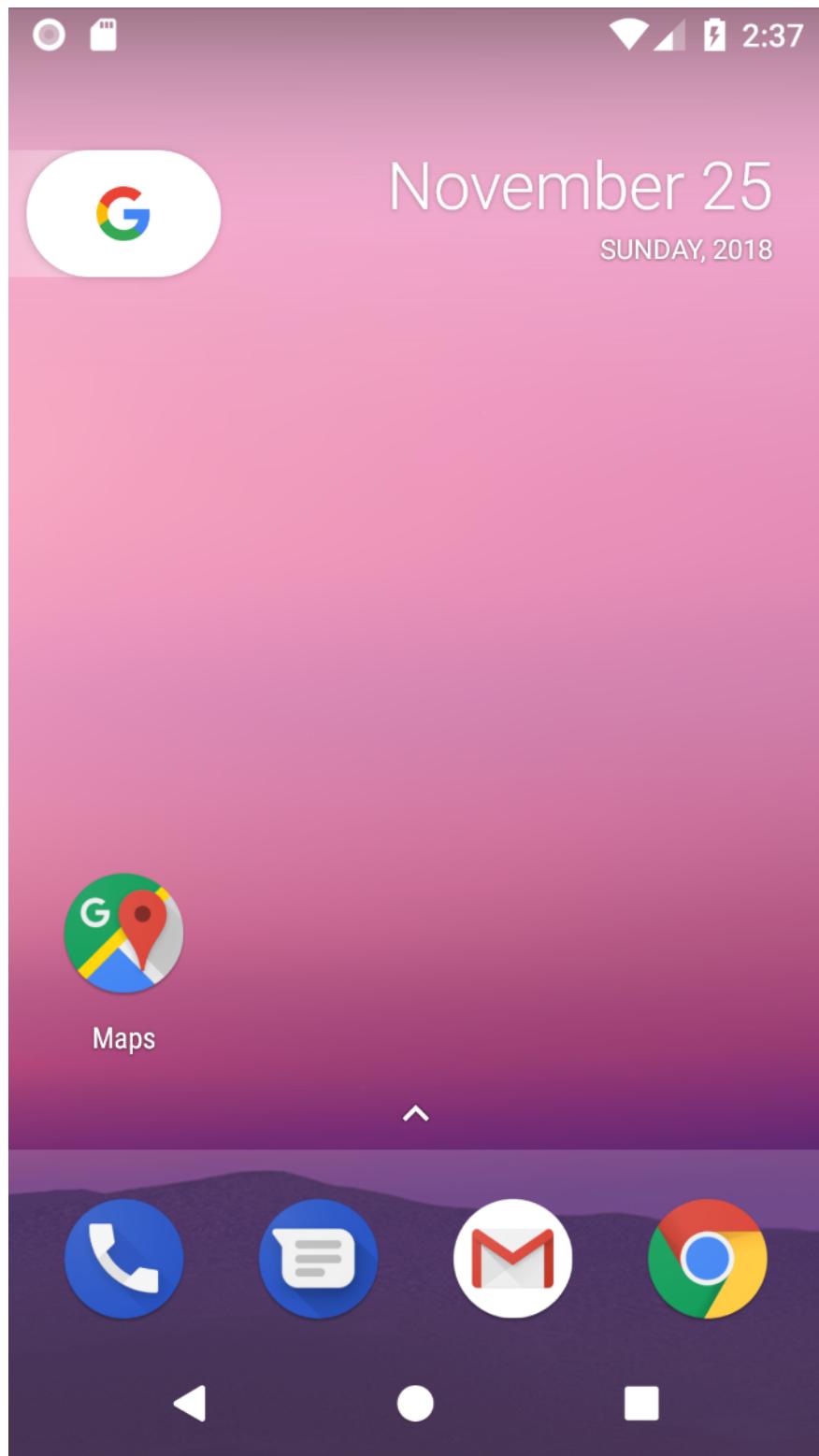
Фиг.54 Настройки за избор на версия на операционната система Android.

Меню **Tools > Android > Android Device Manager** изберете и включете следните настройки:



Фиг.55 Настройки на емулатора за тестване на разработваните приложения.

Тествайте емулатора на операционната система Android натискайки бутона **Start**. Ако всичко е наред ще видите операционната система Android 8.0 Oreo, както е показано на фигурата по-долу:



Фиг. 56. Android 8.0 Oreo

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"
    xmlns:local="clr-namespace:Suma_2._0"
    x:Class="Suma_2._0.MainPage">

    <!-- User Interface (UI): Suma 2.0 -->
    <StackLayout Padding="20">
        <Label Text="Сума2.0" FontSize="Large" />
        <Label Text="A=" />
        <Entry x:Name="EntryA" Keyboard="Numeric" />
        <Label Text="B=" />
        <Entry x:Name="EntryB" Keyboard="Numeric" />
        <Label Text="A+B=" />
        <Entry x:Name="EntryC" Keyboard="Numeric" />
        <Button Text="Изчисли" Clicked="OnButtonClicked" />
    </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 Xamarin.Forms;

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

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

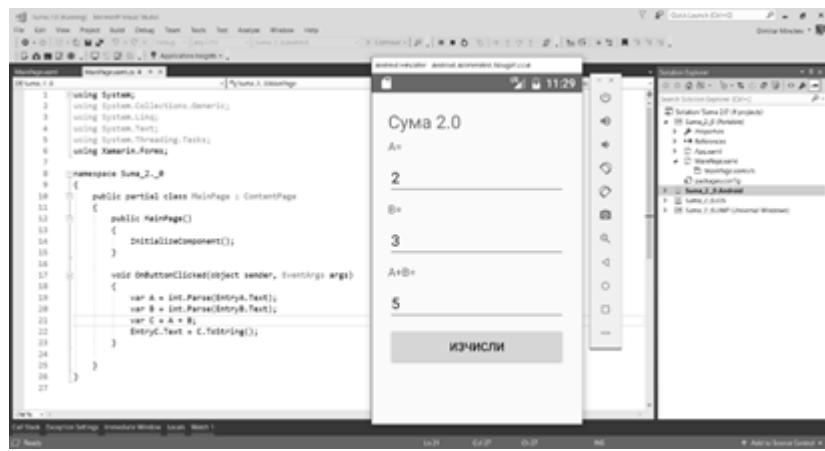
```

Demo

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



Фиг.57 Разработка на мулти-платформено мобилно приложение за намиране сумата на две числа



Фиг.58 Тестване на мулти-платформено мобилно приложение за намиране сумата на две числа - *Android Emulator 7.1 (API 25)*

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 Фибоначи.

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">
        <Label Text="Фибоначи 2.0" FontSize="Large" />
        <Label Text="Ограничение" />
        <Entry x:Name="EntryLimit" Keyboard="Numeric" Text="1000" />
        <ListView x:Name="ListViewNumbers" />
        <Button Text="Генерирай" Clicked="OnButtonClicked" />
    </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;

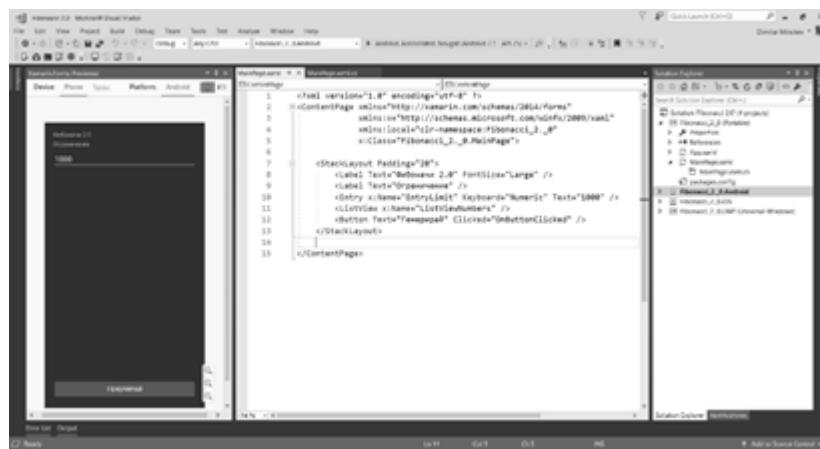
namespace Fibonacci_2._0
{
    // Business Logic (BL): Fibonacci 2.0
    public partial class MainPage : ContentPage
    {
        // Constructor
        public MainPage()
        {
            InitializeComponent();
        }

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

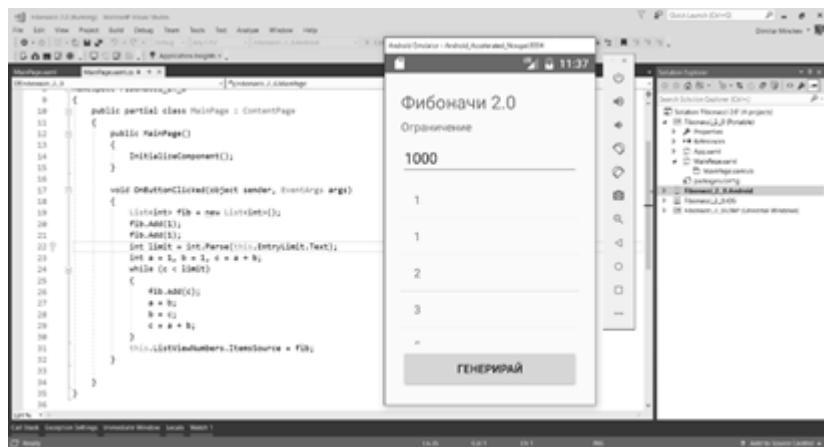
```

Demo

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



Фиг.59 Разработка на мултплатформено мобилно приложение за генериране на редицата от числата на Фибоначи



Фиг.60 Тестване на мултплатформено мобилно приложение за генериране на редицата от числата на Фибоначи - *Android Emulator 7.1 (API 25)*

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.

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">
        <Label Text="Прости 2.0" FontSize="Large" />
        <Label Text="Ограничение" />
        <Entry x:Name="EntryLimit" Keyboard="Numeric" Text="1000" />
        <ListView x:Name="ListViewNumbers" />
        <Button Text="Генерирай" Clicked="OnButtonClicked" />
    </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
{
    // Business Logic (BL): Primes 2.0
    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.EntryLimit.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.ListViewNumbers.ItemsSource = primes;
        }
    }
}

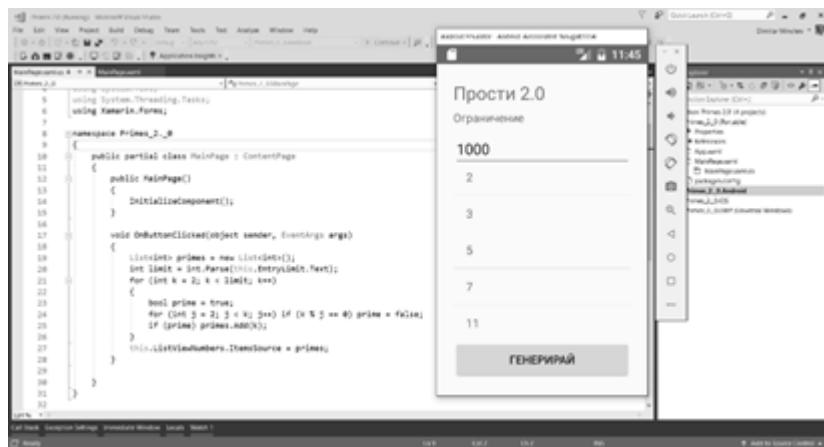
```

Demo

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



Фиг.61 Разработка на мултимедийно мобилно приложение за генериране на редица от прости числа.



Фиг.62 Тестване на мултиплатформено мобилно приложение за генериране на редица от прости числа - Android Emulator 7.1 (API 25)

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">
        <Label Text="Clicker Mania 3.0" FontSize="Large" />
        <Label Text="Timer" />
        <Entry x:Name="Timer" Text="0" />
        <Label Text="Clicks" />
        <Entry x:Name="Clicks" Text="0" />
        <Label Text="Clicks Per Minute" />
        <Entry x:Name="CPM" Text="0" />
        <Button Text="Click" Clicked="OnButtonClicked" />
    </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 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);
        }

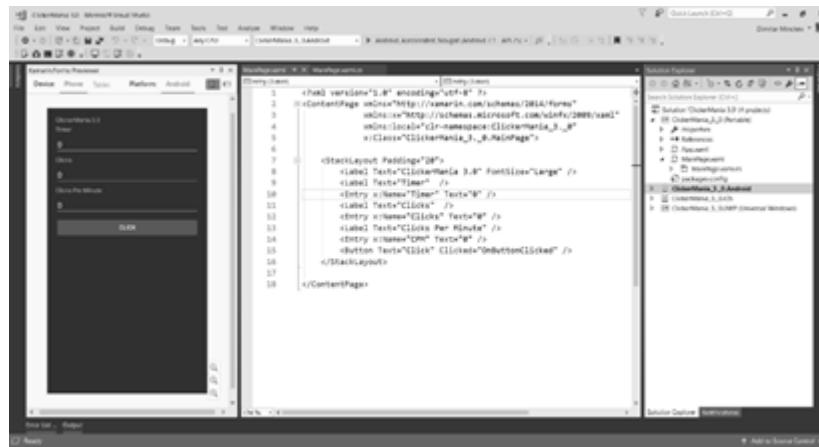
        // Timer Tick Event Handler
        private bool TimerTick()
        {
            int T = int.Parse(this.Timer.Text);
            this.Timer.Text = (++T).ToString();
            // clicks Per Minute
            this.CPM.Text = (float.Parse(this.Clicks.Text) /
                float.Parse(this.Timer.Text)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

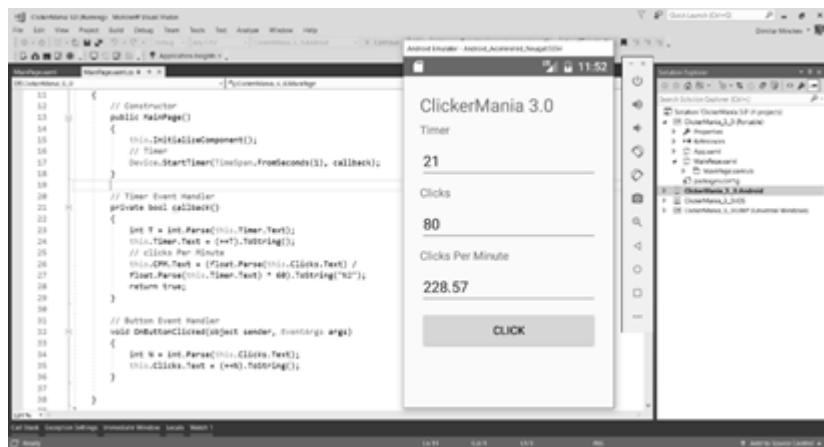
```

Demo

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



Фиг.63 Разработка на мултплатформено мобилно приложение отчитащо броя кликове на потребителя за определено време



Фиг. 64 Тестване на мултиплатформено мобилно приложение отчитащо броя кликове на потребителя за определено време - Android Emulator 7.1 (API 25)

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.

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. Стартирайте 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 0.9.11
```

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">
        <Label Text="HTML Downloader 2.0" FontSize="Medium" />
        <Entry x:Name="URL" Text="http://www.minchev.eu" />
        <Button Text="Download" Clicked="OnButtonClicked" />
        <ScrollView>
            <Label x:Name="HTML" />
        </ScrollView>
    </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 System.Net.Http;
using System.Threading.Tasks;
using Xamarin.Forms;
using AngleSharp.Parser.Html;

namespace HTML_Downloader_2._0
{
    // Business Logic (BL): HTML Downloader 2.0
    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 Go(new Uri(this.URL.Text));

            // Angle Sharp Parsing
            var temp = new HtmlParser().Parse(html);
            string text = temp.Body.TextContent;

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

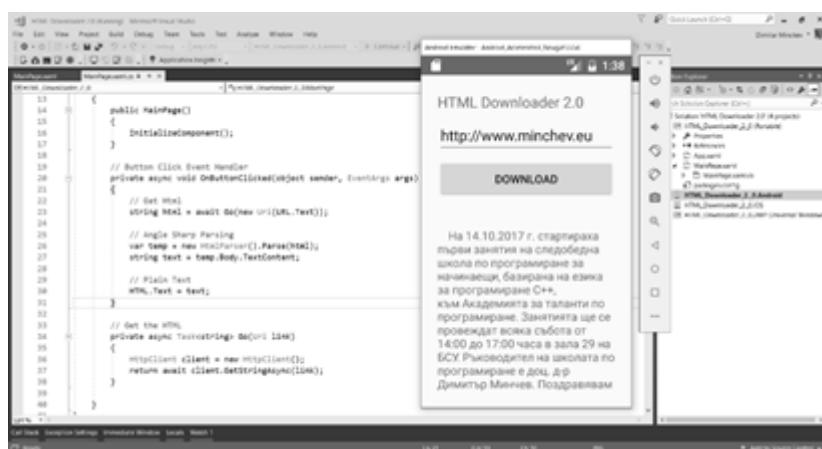
        // Get the HTML code
        private async Task<string> Go(Uri link)
        {
            HttpClient client = new HttpClient();
            return await client.GetStringAsync(link);
        }
    }
}
```

Demo

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



Фиг.65 Разработка на мултиплатформено мобилно приложение за изтегляне HTML съдържанието на кода от Интернет страница.



Фиг.66 Тестване на мултиплатформено мобилно приложение за изтегляне HTML съдържанието на кода от Интернет страница - Android Emulator 7.1 (API 25).

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.

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# > 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 rssFeed = XDocument.Load(url);
            var posts = from item in rssFeed.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;
        }
    }
}

```

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="RSS_Reader_2._0.MainPage">

    <!-- User Interface (UI): RSS Reader 2.0 -->
    <StackLayout Padding="20">
        <Label Text="RSS Reader 2.0" FontSize="Large" />
        <Entry x:Name="EntryURL" Text="https://www.minchev.eu/feed/" />
        <Button Text="Download" Clicked="Button_Clicked" />
        <ListView x:Name="ListViewRSS">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <TextCell Text="{Binding Title}" Detail="{Binding PubDate}" />
                </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;

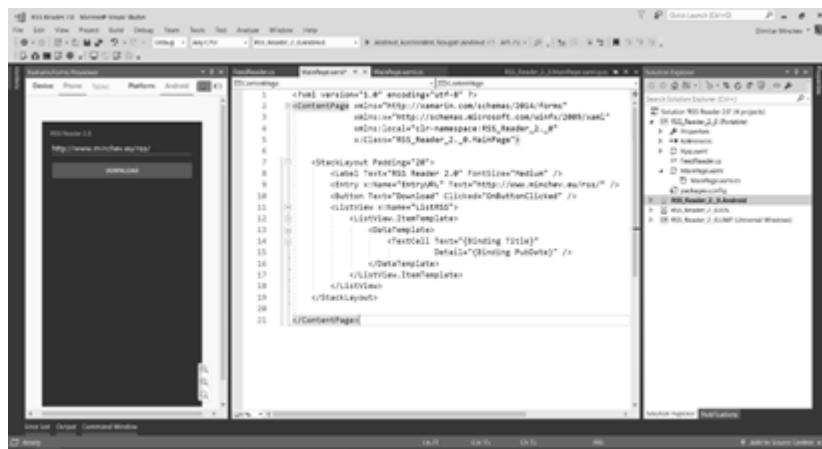
namespace RSS_Reader_2._0
{
    /// <summary>
    /// Business Logic (BL): RSS Reader 2.0
    /// </summary>
    public partial class MainPage : ContentPage
    {
        // Constructor
        public MainPage()
        {
            InitializeComponent();
        }

        // Button Click Event Handler
        private void Button_Clicked(object sender, EventArgs e)
        {
            var link = this.EntryURL.Text;
            var posts = new FeedReader().ReadFeed(link);
            this.ListViewRSS.ItemsSource = new ObservableCollection<FeedItem>(posts);
        }
    }
}

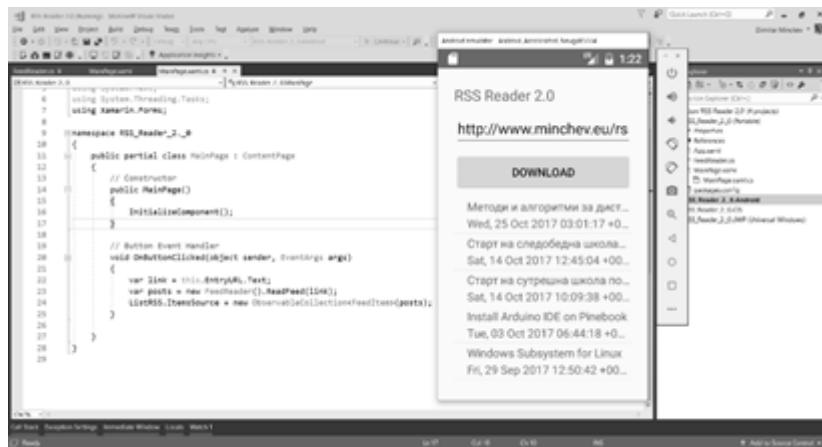
```

Demo

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



Фиг.67 Разработка на мултиплатформено мобилно приложение за четене на Интернет новинарски емисии от RSS източници.



Фиг.68 Тестване на на мултиплатформено мобилно приложение за четене на Интернет новинарски емисии от RSS източници- Android Emulator 7.1 (API 25).

Hyperlinks

Актуализация за добавяне на хипер-връзки по документацията:
[Xamarin.Forms Hyperlinks Microsoft Docs](#)

HyperlinkButton.cs

```

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.Create(
            nameof(Url), typeof(string), typeof(HyperlinkButton), r
        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.cs

```

<?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">
        <Label Text="RSS Reader 2.0" FontSize="Large" />
        <Entry x:Name="EntryURL" Text="https://www.minchev.eu/feed/" />
        <Button Text="Download" Clicked="Button_Clicked" />
        <ListView x:Name="ListViewRSS">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <local:HyperlinkButton Text="{Binding Title}"
                        Detail="{Binding PubDate}"
                        Url="{Binding Link}" />
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
    </StackLayout>
</ContentPage>

```

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**.

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

```
PM> Install-Package Newtonsoft.Json -Version 11.0.2
PM> Install-Package AngleSharp -Version 0.9.11
```

RootObject.cs

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

```
namespace JSON_Reader_2._0
{
    // https://api.chucknorris.io/jokes/random
    // http://json2csharp.com/

    public class RootObject
    {
        public object category { get; set; }
        public string icon_url { get; set; }
        public string id { get; set; }
        public string url { get; set; }
        public string value { 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.

```
<?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 -->
    <StackLayout Padding="20">
        <Label Text="JSON Reader 2.0" FontSize="Medium" />
        <Button Text="Tell Me Joke" Clicked="OnButtonClicked" />
        <ScrollView>
            <Label x:Name="Joke" />
        </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 AngleSharp.Parser.Html;
using Newtonsoft.Json;
using System;
using System.Net.Http;
using Xamarin.Forms;

namespace JSON_Reader_2._0
{
    // Business Logic (BL): JSON Reader 2.0
    public partial class MainPage : ContentPage
    {
        // Constructor
        public MainPage()
        {
            InitializeComponent();
        }

        // Button Event Handler
        async void OnButtonClicked(object sender, EventArgs args)
        {
            // Download JSON
            HttpClient client = new HttpClient();
            var json = await client.GetStringAsync(new Uri("https://api.chucknorris.io"));

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

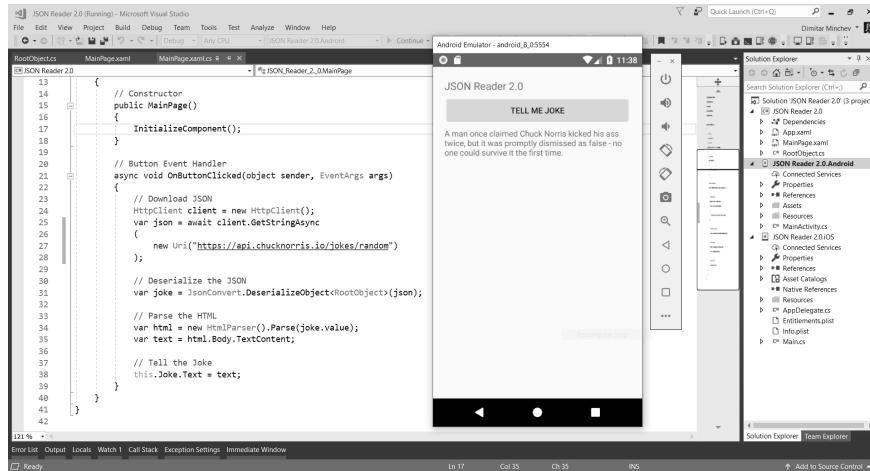
            // Parse the HTML
            var html = new HtmlParser().Parse(joke.value);
            var text = html.Body.TextContent;

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

```

Demo

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



Фиг. 69. Демонстрация на работата на мулти-платформеното мобилно приложение за изтегляне на вицове за Чък Норис

Заключение

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.

Формат и лиценз

Учебното пособие е налично за свободно четене под формата на GitBook базиран [електронен формат](#). Учебните ресурси са налични за свободно изтегляне от GitHub базирано [електронно хранилище](#). Всични учебни материали се разпространяват под безплатен 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. Създава уникалните [Академията за таланти по програмиране](#) и [Школа по роботика](#) за ученици от Бургас. Подготвя студенти за [Републиканска студента 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.

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

Пожелавам Ви приятно четене :)

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

1. Инсталирайте следните програмни продукти:

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

2. Стаптирайте PowerShell 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:
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 gitbook-cli  
npm install -g ebook-convert  
gitbook install
```

5. Възможно е да получите подобна грешка:

```
Installing GitBook 3.2.3  
C:\Users\mitko\AppData\Roaming\npm\node_modules\gitbook-cli\node_modules\npm\node_modu
    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 (fs.js:193:5)
```

За да поправите тази грешка, отворете и редактирайте следния файл:

```
C:\Users\mitko\AppData\Roaming\npm\node_modules\gitbook-cli\node_modules\npm\node_mod
```

Коментирайте редовете от 62 до 64, както е показано по-долу:

```
// fs.stat = statFix(fs.stat)
// fs.fstat = statFix(fs.fstat)
// fs.lstat = statFix(fs.lstat)
```

6. Върнете се в PowerShell 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:

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

Резултат от примерно успешно изпълнение ще изглежда така:

```
info: 7 plugins are installed
info: 6 explicitly listed
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 29 pages
info: found 100 asset files
info: >> generation finished with success in 21.7s !
info: >> 1 file(s) generated
```

Последни думи

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