



Eötvös Loránd Tudományegyetem
Informatikai Kar

Eseményvezérelt alkalmazások

10. előadás

Bevezetés az Avalonia UI keretrendszerbe

Cserép Máté
mcserep@inf.elte.hu
<https://mcserep.web.elte.hu>

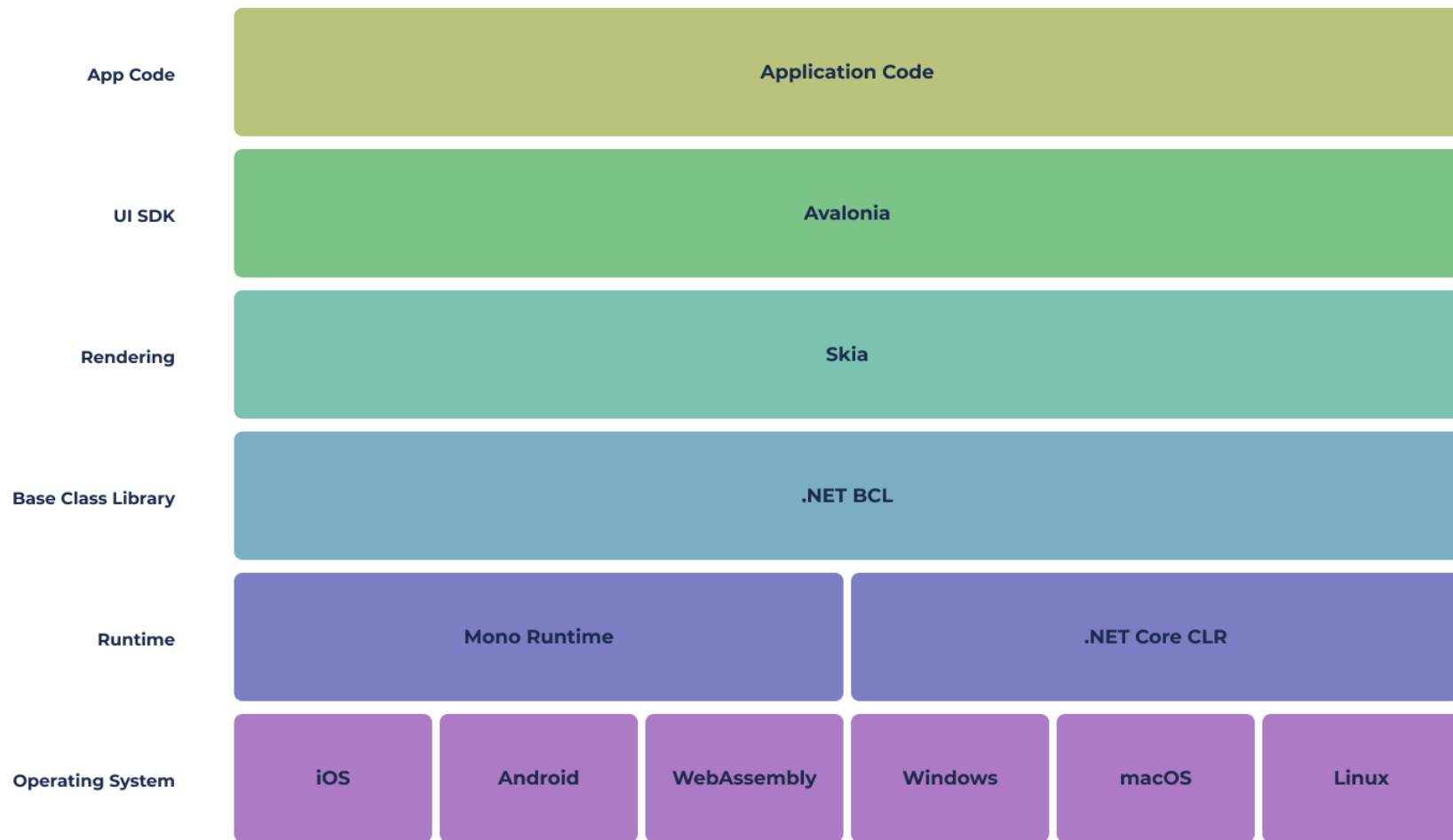
Avalonia UI alapismeretek

Tulajdonságai

- Az *Avalonia UI* egy a .NET környezetre épülő, multi-platform keretrendszer eseményvezérelt grafikus alkalmazások fejlesztésére
 - Windows, Linux, macOS, Android, iOS platformok támogatása, továbbá WebAssembly támogatás
 - alapja a .NET keretrendszer, amely cross-platform fejlesztést tesz lehetővé a támogatott platformok között
 - támogatja a fejlesztést MV architektúrában is, de kifejezetten az MVVM architektúra használata javasolt
 - lehetőséget ad a felület deklaratív leírására (*XAML*)
 - nézetmodellje kompatibilis a WPF és MAUI alkalmazások nézetmodell rétegével (megfelelő tervezés esetén)
 - opcionálisan lehetővé teszi a *reaktív programozást*

Avalonia UI alapismeretek

Architektúra



Avalonia Architecture

Avalonia UI alapismeretek

Telepítés

- Az Avalonia NuGet csomagok telepítésén keresztül könnyedén egy létező projekthez adható, azonban a kezdeti fejlesztéskor hasznos lehet az előre készített sablon projektek (*template*-ek) használata.

- Telepítés parancssorból:

```
dotnet new install Avalonia.Templates
```

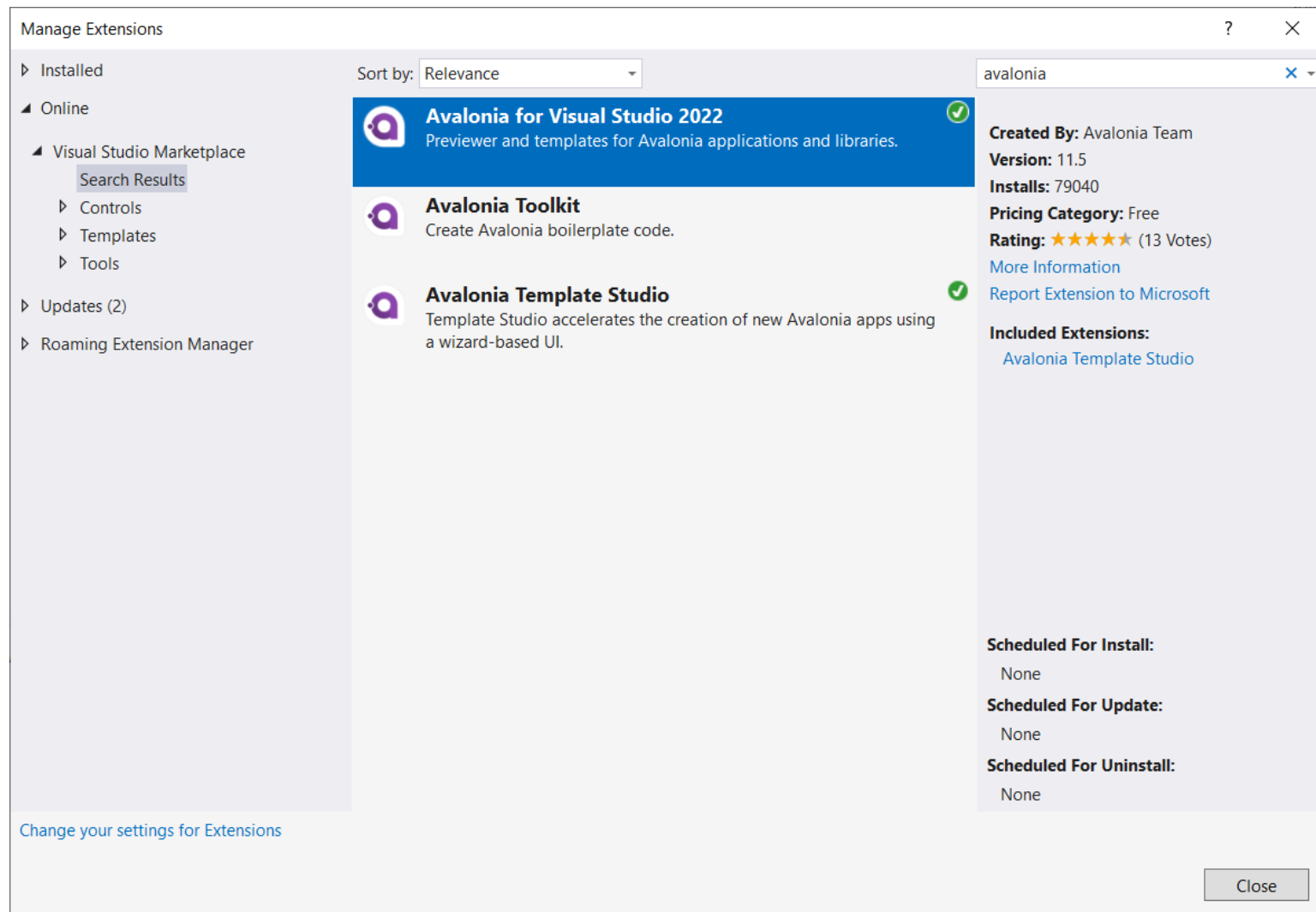
- Majd új MVVM alkalmazás létrehozása:

```
dotnet new avalonia.mvvm -o MyApp -n MyApp
```

- Visual Studio használata esetén célszerű az *Avalonia for Visual Studio 2022* telepítése, így nem csak a sablonok, hanem a grafikus felület tervezőt is használhatjuk.
 - JetBrains Riderhez is elérhető hasonló támogatás.

Avalonia UI alapismeretek

Telepítés



Avalonia UI alapismeretek

Telepítés

- Android platformra fejlesztés támogatásához a következőket telepítenünk szükséges az Android SDK-t (opcionálisan emulátort), valamint a Java SDK-t is
 - A legegyszerűbb megoldás, ha a *MAUI workload*-ot telepítjük a Visual Studio telepítőjéne újrafuttatásával
 - Linux alatt a **dotnet workload install android** parancs kiadásával telepíthető a minimálisan szükséges workload, majd Android SDK és Java SDK külön telepítése szükséges még
- WebAssembly fordítás támogatásához a *WebAssembly Build Tools* csomag telepítése szükséges a *Web Development workload* nem kötelező csomagjai közül





Avalonia UI alapismeretek

Mobil fejlesztés támogatásának telepítése Visual Studioban





Modifying — Visual Studio Enterprise 2022 — 17.4.1

Workloads Individual components Language packs Installation locations

Web & Cloud (4)

-  **ASP.NET and web development** ☒
Build web applications using ASP.NET Core, ASP.NET, HTML/JavaScript, and Containers including Docker supp...
-  **Azure development** ☐
Azure SDKs, tools, and projects for developing cloud apps and creating resources using .NET and .NET Framework....
-  **Python development** ☐
Editing, debugging, interactive development and source control for Python.
-  **Node.js development** ☐
Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Desktop & Mobile (5)

-  **.NET Multi-platform App UI development** ☒
Build Android, iOS, Windows, and Mac apps from a single codebase using C# with .NET MAUI.
-  **.NET desktop development** ☒
Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET and .NET Frame...
-  **Desktop development with C++** ☐
Build modern C++ apps for Windows using tools of your choice including MSVC, Clang, CMake, or MSBuild.
-  **Universal Windows Platform development** ☒
Create applications for the Universal Windows Platform with C#, VB, or optionally C++.

Location
C:\Program Files\Microsoft Visual Studio\2022\Enterprise

By continuing, you agree to the [license](#) for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the [3rd Party Notices](#) or in its accompanying license. By continuing, you also agree to those licenses.

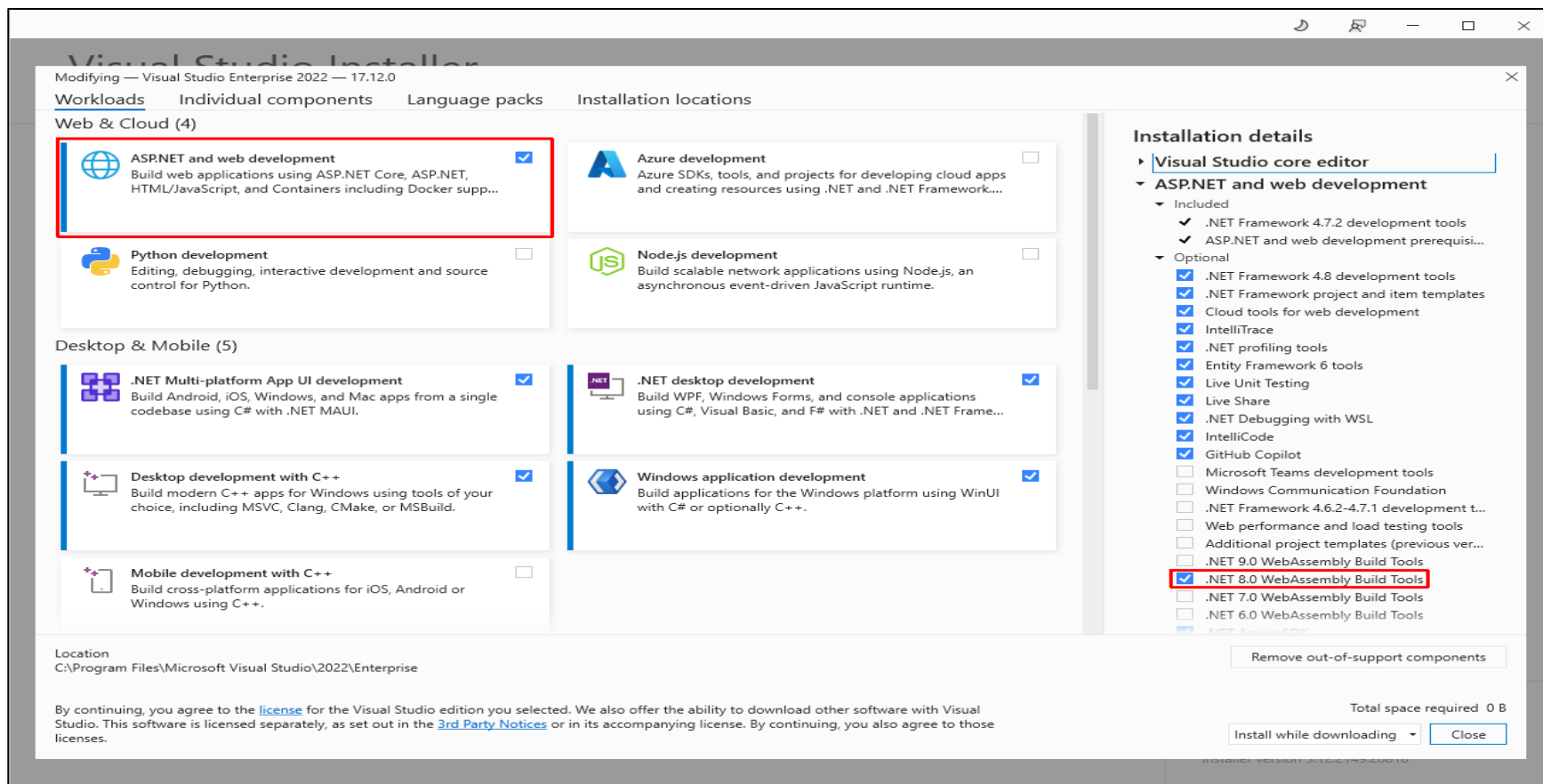
Remove out-of-support components

Total space required 0 B

Install while downloading

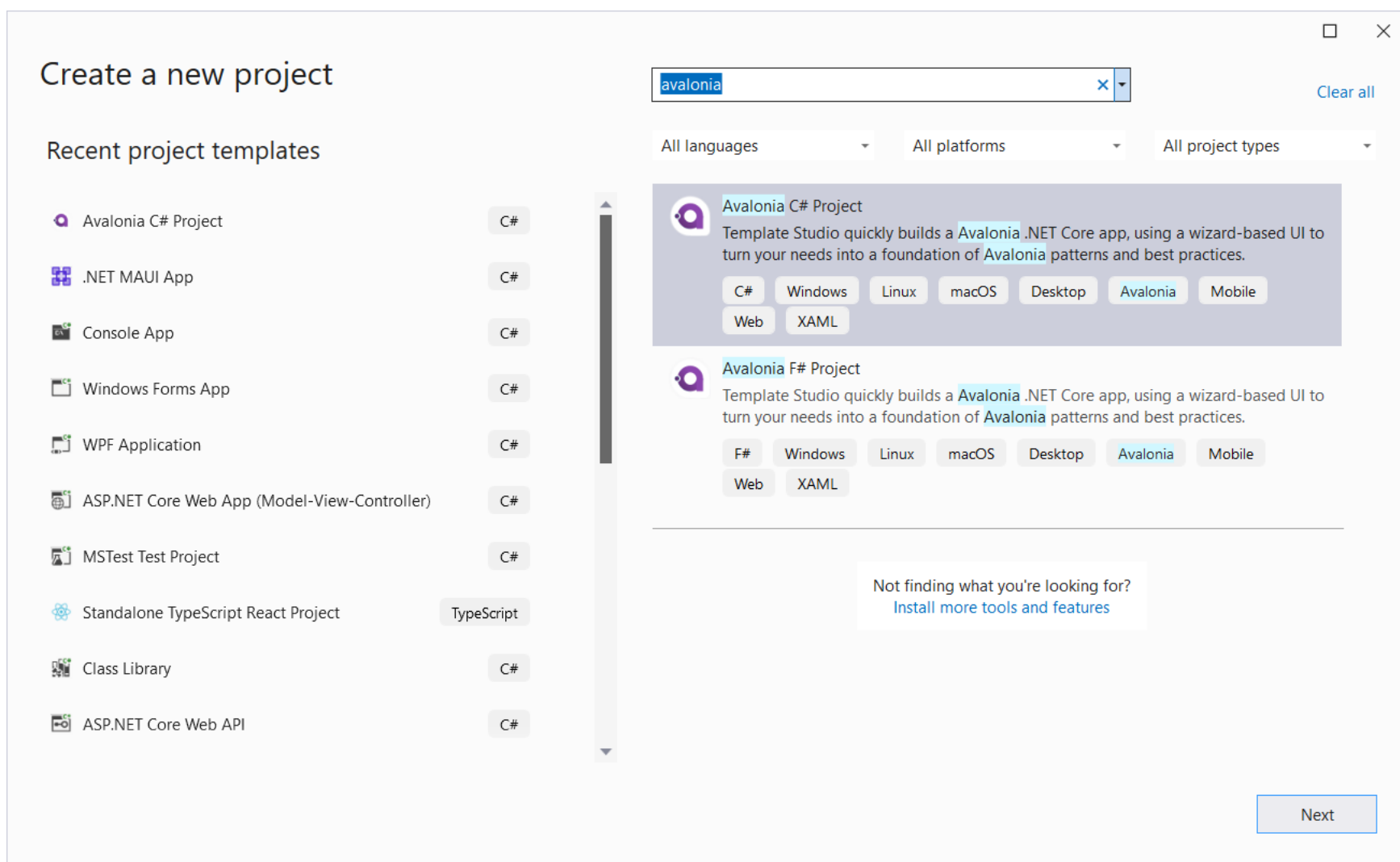
Avalonia UI alapismeretek

WebAssembly fejlesztés támogatásának telepítése Visual Studioban



Avalonia UI alapismeretek

Új projekt létrehozása



Avalonia UI alapismeretek

Új projekt létrehozása


New Avalonia app (AvaloniaApplication1)

1. Platform ✓


2. Design pattern

3. Features


Select target platforms




Desktop
This option allows you to run your project on Windows, OSX, Linux.
[Details](#)



Web
This option allows you to run your project on Web using Web Assembly.
[Details](#)




Android
This option allows you to run your project on Android devices.
[Details](#)




iOS
This option allows you to run your project on iOS devices.
[Details](#)


Your project details

Target platforms

 Desktop ×

 Android ×

Design pattern

 Community Toolkit ▾

About

[About Template Studio](#)
[Report issue](#)
Wizard version: 1.3

By continuing, you agree to the terms of the above licenses.

Back

Next

Create

Cancel

ELTE IK, Eseményvezérelt alkalmazások

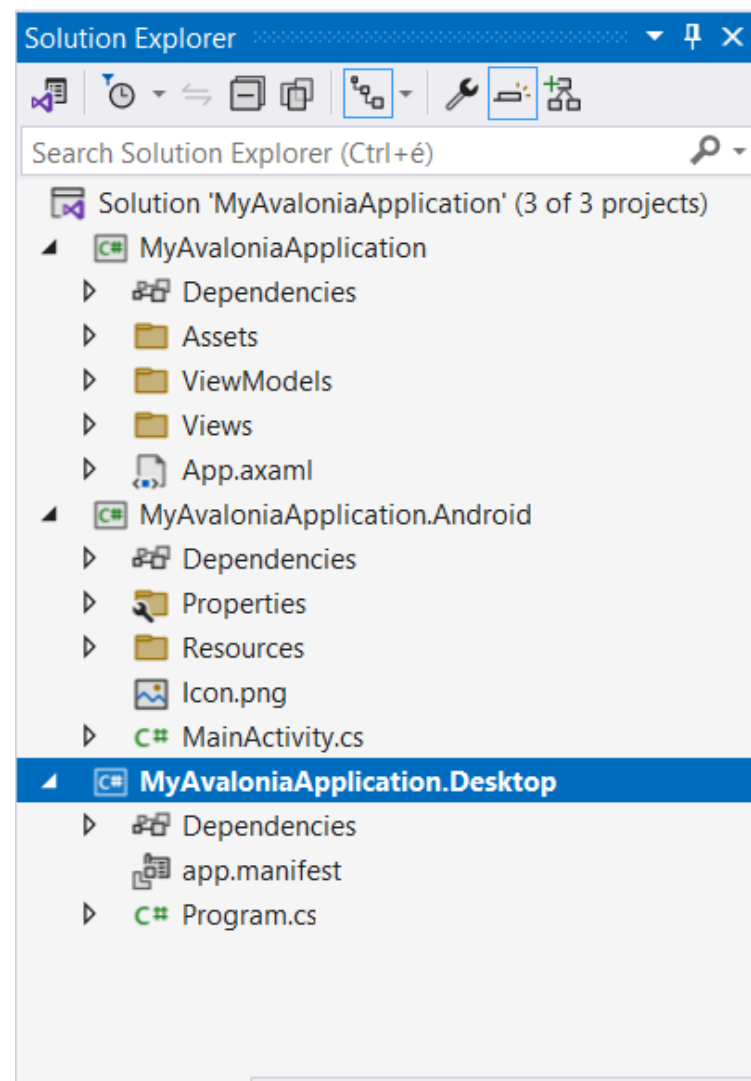
10:10

Avalonia UI alapismeretek

Felépítés

Multi-projekt szerkezet

- *.NET Class Library* projekt a cross-platform kódbázisnak
- Platformfüggő projektek (végrehajtható bináris kimenettel) a platformspecifikus kódrészeknek



Avalonia UI alapismeretek

Avalonia alkalmazások felépítése

- Avalonia UI alkalmazások esetén a közös programegységek (osztálykönyvtárak) tartalmazzák
 - a modellt, amely tartalmazza az üzleti logikát, szokványos eszközök segítségével felépítve
 - a nézetmodellt, amelyet az alapvető eszközök segítségével tudunk felépíteni (**ICommand**, **INotifyPropertyChanged**, stb.)
 - a nézetet, amely XAML alapon írunk le, és adat- és parancskötés (**Binding**) segítségével kapcsoljuk a nézetmodellhez
 - az alkalmazás vezérlését (**App**), amely meghatározza a közös viselkedést minden platformon
 - a cross-platform perzisztenciát; vagy egyedi perzisztencia esetén annak interfészét, amely megvalósítása platformonként eltérhet

Avalonia UI alapismeretek

Avalonia alkalmazások felépítése

- Az alkalmazások közös funkcionalitását *.NET Standard Library* segítségével valósíthatjuk meg
 - a közös programegységek is felruházhatóak platformspecifikus jellemzőkkel (kondicionális fordítással vagy a platform futási idejű vizsgálatával)
- A platformspecifikus programegységek (Desktop, Android, iOS, WebAssembly) tovább bővíthetik a közös funkcionalitást
 - tartalmazhatnak egyedi perzisztencia megvalósítást, mivel az adattárolás módja platformonként eltér
 - tartalmazhatnak speciális nézetbeli elemeket, amelyek adott platformban érhetőek csak el, illetve lehetőséget a nézet adaptálására

Avalonia UI alapismeretek

Avalonia grafikus felület felépítése

- Az Avalonia UI alkalmazások egy egységes, *Google Skia* alapú grafikus felülettel rendelkeznek
 - a felületet deklaratív módon, XAML szintaxissal írhatjuk le
 - a fájlok kiterjesztése **.axaml**, de ennek csak technikai oka van (Visual Studio integráció)
 - a felület platformtól függően ablakokból (asztali alkalmazás) vagy nézetekből (mobil alkalmazás) áll
 - amennyiben asztali- és mobilplatformot is támogatunk, a felületet a nézetekben (**UserControl**) valósítjuk meg, ezeket ágyazzuk közvetlenül az ablakokba (**Window**)
 - így elkerülhető a kódredundancia a nézet rétegben

Avalonia UI alapismeretek

Avalonia grafikus felület felépítése

- Pl. (MainView.axaml):

```
<UserControl xmlns=https://github.com/avaloniaui
    ...
    x:Class="MyApp.Views.MainView">

    <TextBlock Text="Hello Avalonia UI!"
        HorizontalAlignment="Center"
        VerticalAlignment="Center" />

</UserControl>
```


Avalonia UI alapismeretek

Avalonia grafikus felület felépítése

- Pl. (MainWindow.axaml):

```
<Window xmlns="https://github.com/avaloniaui"
    ...
    xmlns:views="clr-namespace:MyApp.Views"
    x:Class="MyApp.Views.MainWindow">

    <views:MainView />

</Window>
```

Avalonia UI alapismeretek

Avalonia grafikus felület felépítése

- Az Avalonia UI felületi vezérlői és lehetőségeik ismerősek lehetnek a WPF keretrendszerből, de eltérések adódnak.
 - Széles körű beépített vezérlő áll rendelkezésre:
<https://docs.avaloniaui.net/docs/basics/user-interface/controls/builtin-controls>
 - Megjelenítők (**TextBlock**, **Label**, **AutoCompleteBox**, stb.)
 - Nyomógombok (**Button**, **ToggleButton**, **RadioButton**, stb.)
 - Csoportos megjelenítők (**ListBox**, **ItemsControl**, stb.)
 - Beviteli vezérlők (**TextBox**, **Slider**, **Calendar**, stb.)
 - Elrendezők (**Canvas**, **Grid**, **StackPanel**, **WrapPanel**, stb.)

Avalonia UI alapismeretek

Alkalmazás tulajdonságok és kihelyezés

- Az alkalmazások tulajdonságait, képességeit az *alkalmazás leíró* (*application manifest*) segítségével írhatjuk le
 - tartalmazza az alkalmazás nevét, leírását, verzióját és a fejlesztő adatait
 - megadja az engedélyeket a rendszerhez, és más alkalmazásokhoz, pl. internet, kamera, pozicionálás, telefonkönyv, stb.
 - Android esetén az **AndroidManifest.xml** fájl, Windows esetén a **Package.appxmanifest** fájl, iOS/Mac esetén az **Info.plist** fájl tartalmazza a leírást
- A megfelelően konfigurált alkalmazások kihelyezhetőek fizikai eszközökre, illetve elhelyezhetőek a platform alkalmazásboltjában is, ehhez az alkalmazást megfelelő aláírással kell ellátnunk, amely szintén platformspecifikus

Avalonia UI alapismeretek

Példa

Feladat: Készítsünk egy egyszerű számológépet, amellyel a négy alpműveletet végezhetjük el, illetve láthatjuk korábbi műveleteinket is.

- a modell (**CalculatorModel**) biztosítja a számológép funkcionalitást, ezt újrahasznosítjuk
- a nézetben (**MainView**) elhelyezünk egy rácsot, benne a beviteli mezőt (**TextBox**), a gombokat (**Button**), valamint a számítások listáját (**TextBlock**)
- a gombokhoz közös eseménykezelőt rendelünk (**ButtonClicked**), és a gomb szövege alapján döntünk a műveletről
- az esetleges hibákról figyelmeztető üzenetet küldünk (**MessageBox.Avalonia** NuGet csomag használatával)

Avalonia UI alapismeretek

MVVM architektúra

- Az Avalonia UI támogatja az MVVM architektúra alapú fejlesztést, így biztosított
 - az adatkötés (**Binding**) a nézet oldalon, amelynek megadhatunk tetszőleges forrást (**DataContext**)
 - minden vezérlőnek külön is megadható forrás a **DataContext** tulajdonság segítségével
 - az elnevezett elemekre (**x:Name**) is hivatkozhatunk a kötésben (**x:Reference**), pl.:

```
<TextBox x:Name="MyTextBox" />
<Label Text="{Binding
                #MyTextBox.Text.Length}" />
<!-- a címke a szövegdoboz tartalmának
hosszát jelenítse meg -->
```

Avalonia UI alapismeretek

MVVM architektúra

- a változásjelzés (**INotifyPropertyChanged**, **ObservableCollection**), valamint a parancsvégrehajtás (**ICommand**) nézetmodell oldalon
- az alkalmazás vezérlése az alkalmazás (**App** osztály) segítségével
 - a nézet adatforrását a **DataContext** tulajdonságán keresztül adhatjuk meg
 - a nézet kiválasztása azonban már függ az alkalmazás élelciklusától
(**IClassicDesktopStyleApplicationLifetime** vagy **ISingleViewApplicationLifetime**)
 - asztali alkalmazásban tetszőlegesen sok ablakunk lehet, egy fő ablakot kell megadnunk (**MainWindow**)
 - mobil alkalmazásban egy nézet látszódik, ezt kell kicserélni, más felületet megjelenítéséhez (**MainView**)

Avalonia UI alapismeretek

MVVM Toolkit

- MVVM alapú fejlesztést segítő NuGet csomag (*CommunityToolkit.Mvvm*)
 - Használható Avalonia UI-hoz, de WPF-hez is
 - Avalonia UI esetén alapértelmezetten a projekthez adásra került, ha ezt a tervezési mintát választottuk a létrehozáskor
 - Gyakran használt őstípusokat és az MVVM alapú fejlesztést segítő kódgenerátor attribútumokat tartalmaz
- **ViewModelBase: ObservableObject** néven szerepel
 - Továbbra is lehet egy **ViewModelBase** osztályunk ebből származtatva, további közös funkcionalitást biztosítva

Avalonia UI alapismeretek

MVVM Toolkit

- DelegateCommand

- helyette RelayCommand és RelayCommand<T>, pl.:
`public RelayCommand<int> MyCommand { get; set; }`
- metódus attribútumaként is alkalmazható, pl.:

```
[RelayCommand] // SomethingCommand parancs  
private void Something() { ... }
```

- generált kód:

```
private RelayCommand? somethingCommand;  
public IRelayCommand SomethingCommand =>  
    somethingCommand ??= new  
        RelayCommand(Something);
```

Avalonia UI alapismeretek

MVVM Toolkit

- `OnPropertyChanged()`
 - helyette `SetProperty(ref name, value)`, amely beállítja a `name` adattagot, majd kiváltja a `PropertyChanged` eseményt rá
 - `ObservableProperty` adattag attribútumaként, pl.:

```
[ObservableProperty] // Data property  
private string _data { ... }
```

- generált kód:

```
public string? Data {  
    get => name;  
    set => SetProperty(ref _data, value);  
}
```

Avalonia UI alapismeretek

Példa

Feladat: Készítsünk egy egyszerű számológépet, amellyel a négy alapműveletet végezhetjük el, illetve láthatjuk korábbi műveleteinket is.

- valósítsunk meg MVVM architektúrát a nézetmodell kiemelésével
- a nézetmodell (**CalculatorViewModel**) tartalmazza az aktuális értéket (**NumberFieldValue**), a számítások listáját (**Calculations**) és a számítást parancs formájában (**CalculateCommand**)
 - a számítási hibákkal kapcsolatosan eseményt küld (**ErrorOccured**)
- az alkalmazás példányosítja és összeállítja az alkalmazás rétegeit, és kezeli a számítási hibák eseményeit

Avalonia UI alapismeretek

Alkalmazások mobil környezetben

- A mobil/táblagépes környezetben alkalmazásunknak számos speciális követelményt kell teljesíteni
 - *könnyű áttekinthetőség*, infografikus megjelenítés
 - *folyamatos, gyors működés* aszinkron tevékenységekkel
 - *alkalmazkodás (resposiveness)*: az alkalmazás
 - különböző hardvereken,
 - különböző méretű/felbontású képernyőkön,
 - különböző tájolással (portré/tájkép),
 - különböző üzemmódokban (teljes képernyő, oldalra zárt, stb.) futhat
 - *kézmozdulatok* kezelése, és kihasználása
 - *speciális hardverek* igénybevétele (GPS, gyorsulásmérő, stb.)

Avalonia UI alapismeretek

Kézmozdulatok kezelése

- Célszerű az alkalmazás vezérlésében a felhasználó kézmozdulataira támaszkodni
 - bármely vezérlőre állíthatunk kézmozdulat érzékelést (**GestureRecognizer**), így tetszőleges tevékenységet (**Command**) rendelhetünk bármely vezérlőhöz
 - támogatott a görgetés (**ScrollGestureRecognizer**), a csíptetés (**PinchGestureRecognizer**) és a húzás (**PullGestureRecognizer**), illetve tetszőleges egyedi mozdulat megvalósítása (**GestureRecognizer**), pl.:

`<Image ...>`

`<Image.GestureRecognizers> <!-- érzékelés -->`

`<PullGestureRecognizer Command=... />`

`<!-- érintés hatására fut a parancs -->`

Avalonia UI alapismeretek

Méret kezelés

- A mobil eszközök képernyőmérete jelentősen eltérhet, és az alkalmazásunknak alkalmazkodnia kell a teljes képernyős megjelenítéshez
 - használjunk relatív elrendezőket és pozícionálást
 - kezelhető a vezérlő átméretezése (**OnSizeChanged** felüldefiniálásával a nézet háttérkódjában)
 - pl.:

```
protected override void OnSizeChanged(...) {  
    base.OnSizeChanged(e) ;  
  
    Double height = e.NewSize.Height;  
    Double width = e.NewSize.Width;  
    // ...  
}
```

- A mobil eszközök többféle tájolásban helyezkedhetnek el
 - általában az portré/tájkép (álló/fekvő) tájolásokat különböztetjük meg, de ezeknek lehetnek speciális esetei
 - a támogatott tájolásokat mobil platformokon korlátozhatjuk is
 - Android esetén a főtevékenység (**MainActivity**) egyik jellemzője a támogatott tájolás (**ScreenOrientation**)
 - az eszköz tájolását és az alkalmazás képernyőjének méretét egyszerre célszerű szabályozni a vezérlő méretváltoztatásának kezelésével (**OnSizeChanged**)

Avalonia UI alapismeretek

Tájolás kezelés

- pl.:

```
protected override void OnSizeChanged(
    SizeChangedEventArgs e)
    // megkapjuk az aktuális
    // szélességet/magasságot az argumentumban
{
    base.OnSizeChanged(e) ;

    // orientáció meghatározása
    if (e.NewSize.Width > e.NewSize.Height)
        ... // tájkép
    else
        ... // portré
}
```

- Az alkalmazások felhasználhatóak táblagépes, illetve mobil környezetben is, és mindkét környezetben megfelelő megjelenítést kell biztosítanunk
 - kódból lekérhetjük az eszköz típusát (**OperatingSystem**), és arra megfelelően reagálhatunk, pl.:

```
if (OperatingSystem.IsAndroid() ||
    OperatingSystem.IsIOS()) {
    image.Source =
        ImageSource.FromFile("small.jpg");
} else {
    image.Source =
        ImageSource.FromFile("large.jpg");
}
// táblagépen nagyobb képet használunk
```

Avalonia UI alapismeretek

Eszközkezelés

- nézetből az eszköznek megfelelő értékeket adhatjuk át (**OnPlatform**), az alapértelmezett és a specifikus értékek megadásával, pl.:

```
<Label FontSize="{OnPlatform 12, Windows=24}" />
```

- a vezérlők bármely tulajdonsága, akár a tartalmuk is testreszabható ilyen módon pl.:

```
<StackPanel>
```

```
    <OnPlatform>
```

```
        <OnPlatform.Default>
```

```
        ...
```

```
    </OnPlatform.Default>
```

```
    <OnPlatform.iOS>
```

```
    ...
```

```
    </OnPlatform.Windows>
```

```
    </OnPlatform>
```

```
</StackPanel>
```

Avalonia UI alapismeretek

Témák és stílusok

- Az Avalonia UI alkalmazásunk alapvető megjelenítését a beállított téma határozza meg.
 - Jellemzően az alkalmazás szintjén kerül beállításra, pl.:

```
<Application.Styles>  
    <FluentTheme />  
</Application.Styles>
```
- Két beépített téma (**SimpleTheme**, **FluentTheme**) és több NuGet csomagból elérhető is rendelkezésünkre áll.
 - **SimpleTheme**: egyszerű, minimalista téma
 - **FluentTheme**: a Microsoft Fluent Design alkalmazása
 - **Material.Avalonia**: a Google Material Design alkalmazás
 - **Classic.Avalonia**: Windows 9x-re hasonlító téma

Avalonia UI alapismeretek

Témák és stílusok

- Avalonia UI alkalmazás XAML kódjában a stílusok vezérlőkhöz csatolásakor lehetőségünk van CSS-szerű *szelektorok* használatára.
 - Adott típusú vezérlőkre illesztés:
`<Style Selector="Button">`
 - Név alapján konkrét vezérlőre illesztés:
`<Style Selector="#MyButton">`
 - Adott osztállyal (**large**) rendelkező vezérlőkre illesztés:
`<Style Selector="Button.large">`
 - Adott hierarchiának megfelelő vezérlőkre illesztés:
`<Style Selector="StackPanel Button">`
 - Adott tulajdonságokkal rendelkező vezérlőkre illesztés:
`<Style Selector="Button[(Grid.Row)=0]">`

Avalonia UI alapismeretek

Témák és stílusok

- Az osztályokra adatkötés (Binding) is lehetséges, logikai értékek formájában, pl.:

```
<Button ...  
    Classes="Large" <!-- nem kondicionális -->  
    Classes.IsActive="{Binding IsActive}">  
    <!-- kondicionális osztály,  
        adatkötéssel a nézetmodellre -->  
    <Button.Styles>  
        <Style Selector="Button.Large">...</Style>  
        <Style Selector="Button.IsActive">...</Style>  
    </Button.Styles>  
</Button>
```

- A stílusok **UserControl** szintjén is megadhatóak tranzitívan vagy **ResourceDictionary**-ben (hasonlóan WPF-hez).

Avalonia UI alapismeretek

Példa

Feladat: Készítsünk egy egyszerű számológépet, amellyel a négy alpműveletet végezhetjük el, illetve láthatjuk korábbi műveleteinket is.

- alakítsuk át a nézetet, hogy könnyen használható legyen mobil környezetben, és alkalmazkodjon a platformhoz
- szövegbevitel helyett gombokat használunk a számokhoz, ezért ki kell egészítenünk a nézetmodellt a számok kezelésével (**Calculate**)
- a nézetmodellek ősszostályát (**ViewModelBase**) egészítsük ki a tájolás kezelésével (**IsPortrait**, **IsLandscape**), így szelektor osztályok segítségével tudunk reagálni az elforgatásokra
 - ehhez a nézet **OnSizeChange()** műveletét felül kell definiálnunk