



Eötvös Loránd Tudományegyetem
Informatikai Kar

Eseményvezérelt alkalmazások

10. előadás

MAUI alapismeretek

Cserép Máté

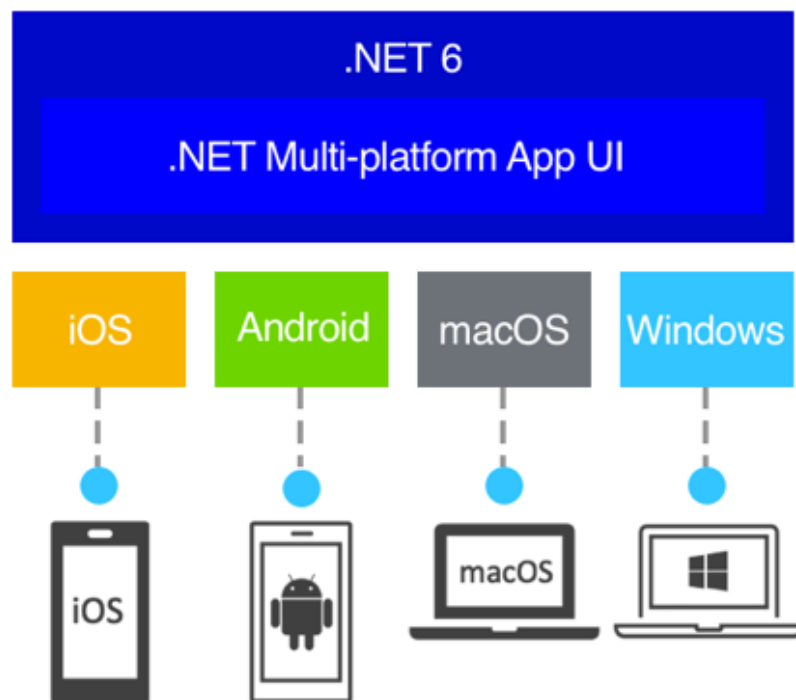
mcserep@inf.elte.hu

<https://mcserep.web.elte.hu>

MAUI alapismeretek

A MAUI platform

- A *MAUI* egy többplatformos szoftverfejlesztői környezet, amely lehetőséget ad Android, iOS, Windows (*WinUI 3* könyvtárral) és macOS (*Mac Catalyst* könyvtárral) alkalmazások fejlesztésére a .NET keretrendszerre alapozva
 - elődje a *Xamarin* keretrendszer (*Xamarin.Forms* folytatása)
 - alapja a .NET 6 keretrendszer, amely *cross-platform* fejlesztést tesz lehetővé a támogatott platformok között



MAUI alapismeretek

A MAUI platform

- A *MAUI* egy többplatformos szoftverfejlesztői környezet, amely lehetőséget ad Android, iOS, Windows (*WinUI 3* könyvtárral) és macOS (*Mac Catalyst* könyvtárral) alkalmazások fejlesztésére a .NET keretrendszerre alapozva
 - elődje a *Xamarin* keretrendszer (*Xamarin.Forms* folytatása)
 - alapja a .NET 6 keretrendszer, amely *cross-platform* fejlesztést tesz lehetővé a támogatott platformok között
 - lehetőséget ad közös kódbázissal rendelkező alkalmazások fejlesztésére, amelyek köztes nyelven (.NET Runtime-mal) vagy natív alkalmazásként futtathatóak (iOS esetében)
 - egységesíti az architektúrát (MVVM) és a grafikus felületet is
 - programcsomagok NuGet segítségével kezelhetők és frissíthetőek

MAUI alapismeretek

MAUI alkalmazások felépítése

- MAUI 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 adatkötés (**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

MAUI alapismeretek

MAUI alkalmazások felépítése

- Az alkalmazások közös funkcionalitását *.NET Standard Library* segítségével valósíthatjuk meg
 - a közös programegységek is felruházhatóak platformspecifikus jellemzőkkel (**Device**)
- A platformspecifikus programegységek (Android, iOS, UWP) 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 (pl. Material Design)





MAUI alapismeretek

Telepítés 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...
-  **Python development**
Editing, debugging, interactive development and source control for Python.
-  **Azure development**
Azure SDKs, tools, and projects for developing cloud apps and creating resources using .NET and .NET Framework...
-  **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

Remove out-of-support components

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.

Total space required 0 B

Install while downloading Close

MAUI alapismeretek

MAUI projekt létrehozása Visual Studioban

The image shows the Visual Studio interface for creating a new project. On the left, the 'Create a new project' window is open, displaying a list of recent project templates. The templates include:

- .NET MAUI App (C#)
- WPF Application (C#)
- Console App (C#)
- Console App (.NET Framework) (C#)
- Windows Forms App (C#)
- MSTest Test Project (C#)
- NUnit Test Project (C#)
- xUnit Test Project (C#)
- Class Library (C#)
- Standalone TypeScript React Project (TypeScript)
- ASP.NET Core with React.js (C#)

In the center, a search results window is open, showing the search term 'maui'. The results list the '.NET MAUI App' template, which is highlighted. The description for this template is: 'A project for creating a .NET MAUI application for iOS, Android, Mac Catalyst, WinUI and Tizen'. The supported platforms are listed as: C#, Android, iOS, Mac Catalyst, macOS, MAUI, Tizen, and Windows.

Below the search results, the 'Configure your new project' window is open, showing the configuration options for the selected template. The configuration options are:

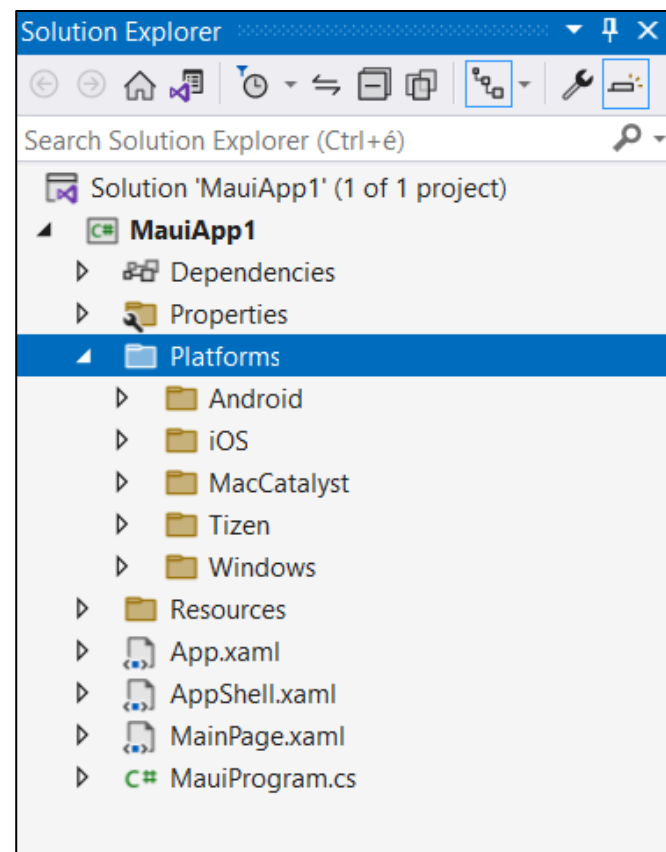
- Project name: MauiApp1
- Location: C:\Users\mcser\source\repos
- Solution name: MauiApp1
- Place solution and project in the same directory:

The 'MAUI' platform is selected in the configuration window. At the bottom right of the configuration window, there are 'Back' and 'Next' buttons.

MAUI alapismeretek

MAUI projekt létrehozása Visual Studioban

- *Single project* paradigma
 - egyetlen projekt fordítható az összes támogatott platformra
 - projekt beállításainál (*Properties*) konfigurálhatóak az egyes támogatni kívánt platformok
 - szemben a Xamarin korábbi *multi project* paradigmájával, ahol egy megosztott közös projekt mellett minden platformnak egy saját projektje is van



MAUI alapismeretek

MAUI alkalmazások felépítése

- A platformfüggetlen belépési pont a **MauiProgram** osztály, amely konfigurálja és elindítja az alkalmazást.
- Pl.:

```
public static MauiApp CreateMauiApp()  
{  
    var builder = MauiApp.CreateBuilder();  
    builder.UseMauiApp<App>()  
        .ConfigureFonts(fonts =>  
        {  
            fonts.AddFont("...");  
        });  
  
    return builder.Build();  
}
```

MAUI alapismeretek

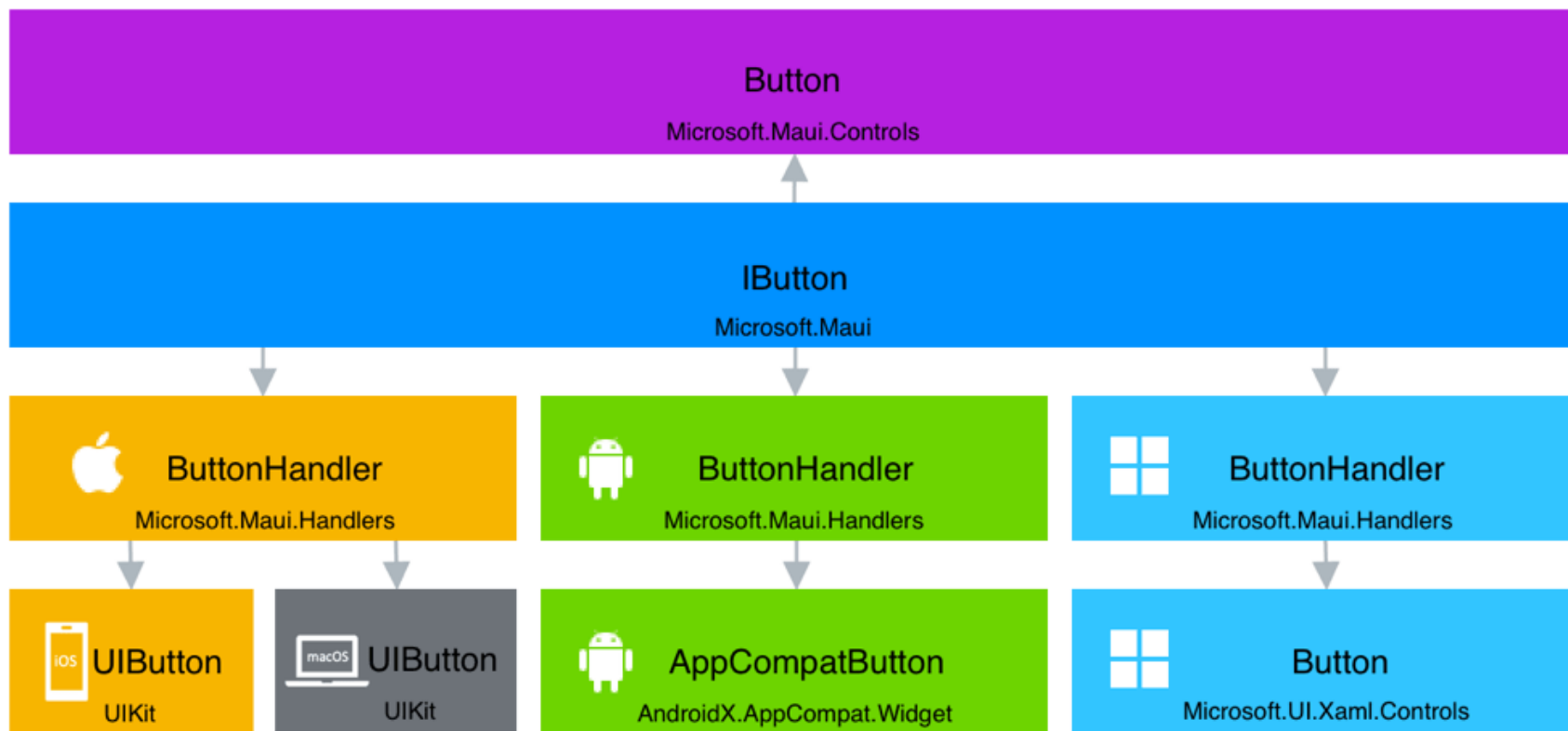
MAUI alkalmazások felépítése

- A platformfüggetlen belépési pont a **MauiProgram** osztály, amely konfigurálja és elindítja az alkalmazást.
- A MAUI alkalmazás keretét és közös erőforrásait az alkalmazás (**Application**) jelenti
 - tartalmazza a nyitóképernyőt (**MainPage**), valamint az alkalmazás életciklus eseménykezelőit
 - kezeli az alkalmazás tulajdonságait (**Properties**), erőforrásait (**Resources**)
- Több képernyős alkalmazások könnyű kezelhetőségét és navigálhatóságát készíti elő az **AppShell** osztály
 - pl. fülek (*tab*) vagy lenyitható menü (*flyout*) használatával
 - így az **AppShell** lesz jellemzően a **MainPage** az alkalmazásban

MAUI alapismeretek

MAUI grafikus felület felépítése

- A MAUI alkalmazások egy egységes grafikus felülettel rendelkeznek, amelyeket a platform egyedi vezérlőire fordít le a keretrendszer



MAUI alapismeretek

MAUI grafikus felület felépítése

- A MAUI alkalmazások egy egységes grafikus felülettel rendelkeznek, amelyeket a platform egyedi vezérlőire fordít le a keretrendszer
 - a felület lapokból (**Page**) áll, amelyek különböző felépítéssel rendelkezhetnek
 - pl. egyszerű tartalom (**ContentPage**), többlaposváltozatok (**TabbedPage**, **NavigationPage**, **FlyoutPage**)
 - a lapoknak lehet címe (**Title**), háttere (**BackgroundImage**) és ikonja (**IconImageSource**), valamint platformspecifikus eszköztára (**ToolBarItems**)
 - megjeleníthetnek üzeneteket (**DisplayAlert**), valamint választódialógust (**DisplayActionSheet**)

MAUI alapismeretek

MAUI grafikus felület felépítése

- a tartalmat elrendező (**Layout**) elemekkel igazíthatjuk el, pl. rács (**Grid**), vízszintes/függőleges lista (**StackLayout**), ezek együttese (**FlexLayout**) abszolút pozíciós (**AbsoluteLayout**),
- a tartalmat nézet (**View**) elemekből építhetjük fel, pl. **Label**, **Button**, **DatePicker**, **Entry** (egy soros szövegdoboz), **Editor** (több soros szövegdoboz), **Picker** (kiválasztó), **TableView**, **Image**, **WebView**
 - a nézeten megjelenő betűk alapértelmezetten platformspecifikusak, és mértük is platformnak megfelelően szabályozható (**Small**, **Medium**, stb.)
 - lehetőségünk van egyedileg formázott szöveg megjelenítésére (**FormattedString**)

MAUI alapismeretek

MAUI grafikus felület felépítése

- Pl.:

```
<ContentPage ...>
  <!-- tartalomlap, amely lehet az alkalmazás
        képernyője -->
  <StackLayout Orientation="Vertical">
    <!-- függőleges elrendező -->
    <Label Text="Hello, MAUI!" Margin="5"
          FontSize="Large"/> <!-- címke -->
    <Button Clicked="Button_Clicked"
            Text="Go on.." HorizontalOptions="Center"
            Margin="10" />
    <!-- gomb eseménykezelővel -->
  </StackLayout>
</ContentPage>
```


MAUI 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 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

MAUI alapismeretek

Példa

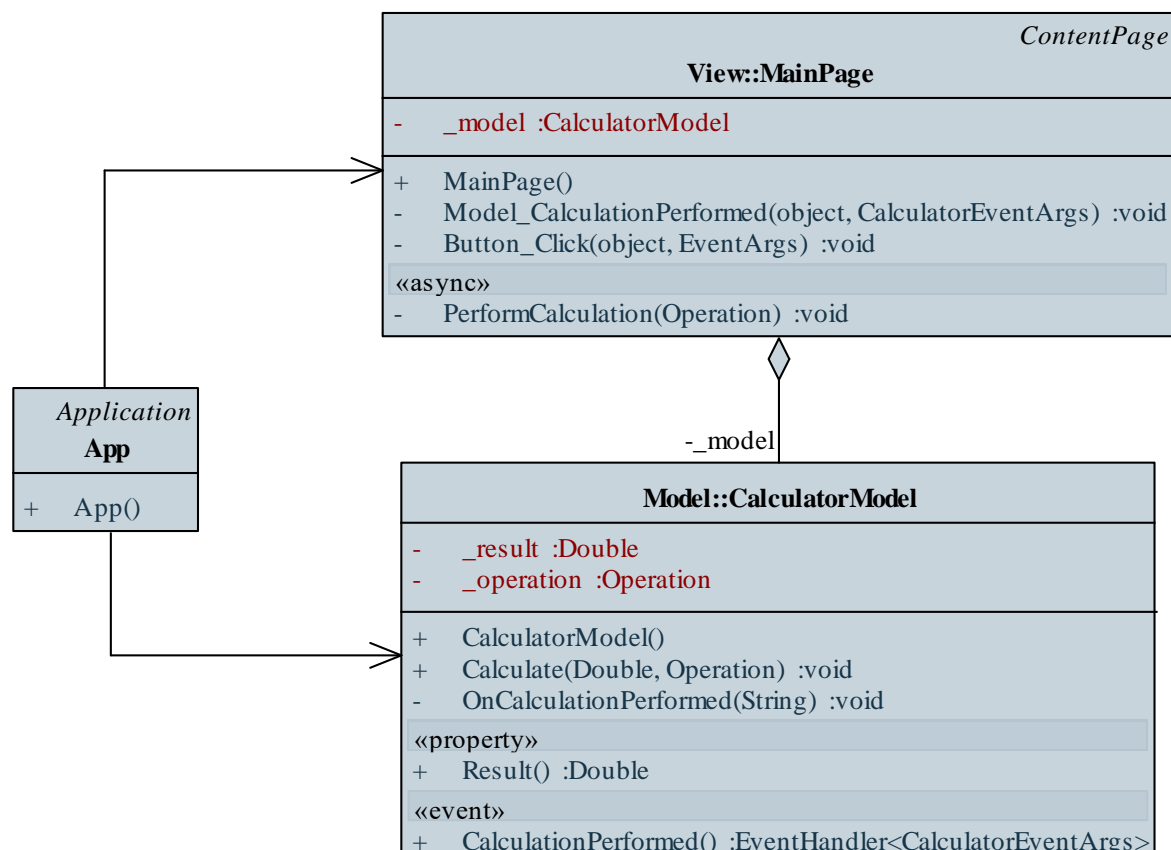
Feladat: Készítsünk egy egyszerű számológépet, amellyel a négy alapműveletet végezhetjük el, illetve láthatjuk korábbi műveleteinket is.

- a modell (**CalculatorModel**) biztosítja a számológép funkcionalitást, ezt újrahasznosítjuk
- a nézetben (**MainPage**) elhelyezünk egy rácsot, benne a beviteli mezőt (**Entry**), a gombokat (**Button**), valamint a számítások listáját (**Label**)
- a gombokhoz közös eseménykezelőt rendelünk (**Button_Blick**), és a gomb szövege alapján döntünk a műveletről
- az esetleges hibákról figyelmeztető üzenetet küldünk (**DisplayAlert**)

MAUI alapismeretek

Példa

Tervezés:



MAUI alapismeretek

MVVM architektúra

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

```
<Label x:Name="someLabel" />
```

```
<Label Text="{Binding  
                Source={x:Reference someLabel},  
                Path=Text}" />
```

```
<!-- a két címke ugyanazt írja ki -->
```

MAUI alapismeretek

MVVM architektúra

- a változásjelzés (**INotifyPropertyChanged**, **ObservableCollection**), valamint a parancsvégrehajtás (**ICommand**, **Command**, **Command<T>**) nézetmodell oldalon
- az alkalmazás vezérlése az alkalmazás (**App** osztály) segítségével
 - a nézet adatforrását a **MainPage** tulajdonság **BindingContext** tulajdonságán keresztül adhatjuk meg, pl.:

```
ViewModel viewModel = new ViewModel(model);  
MainPage = new MainPage();  
MainPage.BindingContext = viewModel;  
// nézetmodell befecskendezése
```
 - a nézet cseréjét a **MainPage** tulajdonságon keresztül végezhetjük, ám ehelyett célszerű több lapot tartalmazó nézet (pl. **TabbedPage**) vagy **Shell** használata

MAUI alapismeretek

MVVM architektúra

- az **AppShell** használatával könnyedén definiálhatunk például füleket, amelyek nézetei között a felhasználó válthat

```
<Shell ...
  xmlns:views="clr-namespace:MauiApp1.Views">
  <TabBar>
    <ShellContent Title="Main"
      Icon="main.png"
      ContentTemplate="{DataTemplate
        views:MainPage}" />
    <ShellContent Title="Settings"
      Icon="settings.png"
      ContentTemplate="{DataTemplate
        views:SettingsPage}" />
  </TabBar>
</Shell>
```


MAUI alapismeretek

Példa

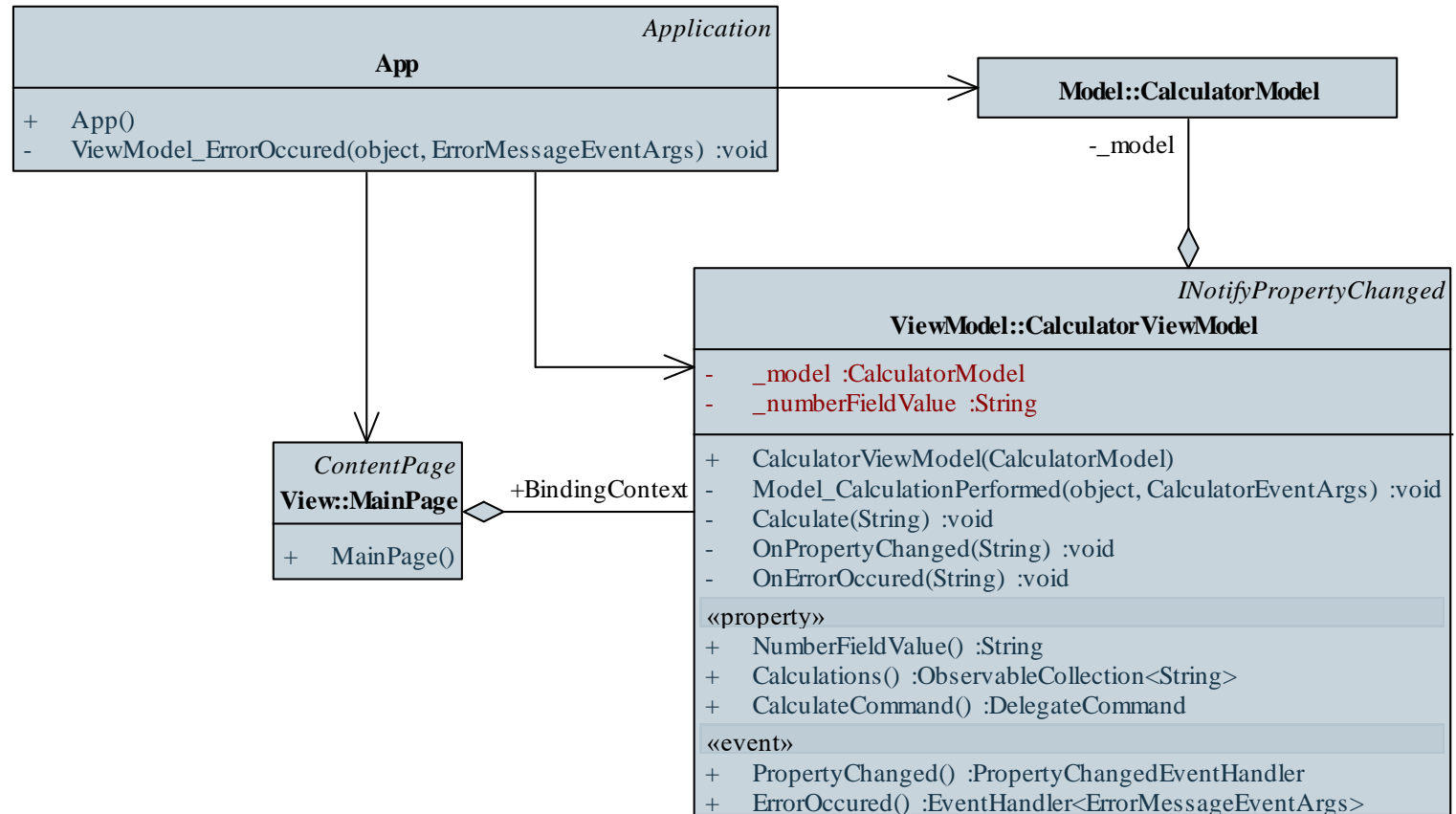
Feladat: Készítsünk egy egyszerű számológépet, amellyel a négy alapműveletet végezhetjük el, illetve láthatjuk korábbi műveleteinket is.

- 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

MAUI alapismeretek

Példa

Tervezés:



MAUI alapismeretek

Példa

Megvalósítás (MainPage.xaml):

...

```
<!-- felhelyezzük a vezérlőket a rácsra, és  
      hozzákötjük őket a nézetmodellhez -->
```

```
<Entry Text="{Binding NumberFieldValue}"
```

```
      Grid.Row="0" Grid.Column="0"
```

```
      Grid.ColumnSpan="3" FontSize="40" />
```

```
<Button Command="{Binding CalculateCommand}"
```

```
      CommandParameter="+" Text="+"
```

```
      FontSize="30" HeightRequest="60"
```

```
      WidthRequest="70" HorizontalOptions="Start"
```

```
      Grid.Row="1" Grid.Column="0" />
```

...

MAUI 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.)

MAUI 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 az érintés (**TapGestureRecognizer**), a csíptetés (**PinchGestureRecognizer**) és a húzás (**PanGestureRecognizer**), illetve tetszőleges egyedi mozdulat megvalósítása (**IGestureRecognizer**), pl.:

```
<Label ...>
```

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

```
    <TapGestureRecognizer Command=... />
```

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

MAUI alapismeretek

Méret kezelés

- A mobil eszközök képernyőmérete jelentősen eltérhet, és az alkalmazásunknak alkalmazkodnia kell a teljes képernyős megjelenítéshez
 - a vezérlők, szövegek méretezésénél használjunk relatív méreteket, pl.:

```
<StackLayout HorizontalOptions="Fill"
                VerticalOptions="Fill" ...>
  <!-- az elrendező kitölti a képernyőt -->
  <Label ... FontSize="Medium">
  <!-- a szövegméret közepes lesz -->
```
- lehetőségünk méretezni bármely vezérlőt a **Scale** tulajdonság segítségével

MAUI alapismeretek

Méret kezelés

- a méretezést legcélszerűbb gyerekvezérlőkre alkalmazni (**ContentView**, **Layout**, **ContentPage** leszármazottakban)
 - kezelhető a vezérlő átméretezése (**OnSizeAllocated**)
 - kezelhető a tartalom átméretezése (**Content.SizeChanged**)
- pl.:

```
private void Content_SizeChanged(...) {  
    // ha változik a tartalom mérete, akkor  
    // méretezzük  
    Double height = Height / Content.Height;  
    Double width = Width / Content.Width;  
  
    if (width > 0 && height > 0)  
        Content.Scale = Math.Min(height, width);  
}
```

MAUI alapismeretek

Tájolás kezelés

- A mobil eszközök többféle tájolásban helyezkedhetnek el
 - általában az portré/tájkép (álló/fekvő) tájolásokat különböztetjük meg, de ezeknek lehetnek speciális esetei
 - a támogatott tájolásokat eszközönként adhatjuk meg (korlátozhatjuk)
 - Android esetén a főtevékenység (**MainActivity**) egyik jellemzője a támogatott tájolás (**ScreenOrientation**)
 - WinUI esetén az alkalmazás leírója (*application manifest*) tartalmazza a tájolást
 - az eszköz tájolását és az alkalmazás képernyőjének méretét egyszerre célszerű szabályozni a vezérlő méretváltogatásának kezelésével (**OnSizeAllocated**)

MAUI alapismeretek

Tájolás kezelés

- pl.:

```
protected override void OnSizeAllocated(
    Double width, Double height)
    // megkapjuk az aktuális
    // szélességet/magasságot
{
    base.OnSizeAllocated(width, height);

    // orientáció meghatározása
    if (width > height)
        ... // tájkép
    else
        ... // portré
}
```

MAUI alapismeretek

Eszközkezelés

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

```
if (DeviceInfo.Idiom == TargetIdiom.Phone) {  
    image.Source =  
        ImageSource.FromFile("small.jpg");  
} else {  
    image.Source =  
        ImageSource.FromFile("large.jpg");  
}  
// táblagépen nagyobb képet használunk
```

MAUI alapismeretek

Eszközkezelés

- nézetből az eszköznek megfelelő értékeket adhatjuk át (**OnIdiom**), ehhez meg kell adnunk az érték típusát (**x:TypeArguments**), illetve a két változatot (**Phone**, **Tablet**), pl.:

```
<Label ...>
  <Label.FontSize>
    <!-- a betűméret eszközfüggő lesz -->
    <OnIdiom x:TypeArguments="x:Double"
      Phone="10" Tablet="30"/>
    <!-- telefon esetén 10-es betűméretet
      használunk, táblagép esetén
      30-ast -->
  </Label.FontSize>
</Label>
```

MAUI alapismeretek

Eszközkezelés

- A **DeviceInfo** osztályon keresztül további információkat is nyerhetünk az eszköztől, ha szükségünk van rá
 - a **DeviceInfo.Platform** a használt platformot kérdezi le (Windows, Android, iOS, macOS, stb.)
 - a **DeviceInfo.DeviceType** megadja, hogy fizikai vagy virtuális eszközön fut-e az alkalmazás
 - a **DeviceInfo.Model**, **DeviceInfo.Manufacturer**, **DeviceInfo.Name** és **DeviceInfo.VersionString** megadja az eszköz gyártóját, a modellt, az operációs rendszert és verziószámát
- A **DeviceDisplay** osztály a képernyőről ad információt, például a **DeviceDisplay.MainDisplayInfo.Orientation** is megadja az orientációt, de lekérhető a felbontás és a pixelsűrűség is.

MAUI alapismeretek

Triggerek

- Lehetőségünk van tevékenységeket deklaratív módon, triggerek segítségével megfogalmazni a nézetben
 - triggerek megadhatók vezérlőkben és stílusokban
 - triggerek segítségével reagálhatunk
 - valamely vezérlő tulajdonságának megváltozására (*property trigger*), pl.:

```
<Trigger TargetType="Entry"
          Property="IsFocused" Value="True">
  <!-- ha fókuszbba kerül a szövegdoboz -->
  <Setter Property="BackgroundColor"
          Value="Yellow" />
  <!-- a háttere sárga lesz -->
  ...
```

MAUI alapismeretek

Triggerek

- a kötött adat megváltozására (*data trigger*), pl.:

```
<DataTrigger TargetType="Button"
    Binding="{Binding Count}" Value="0">
    <!-- ha a nézetmodell Count tulajdonsága
    0-ra vált -->
    <Setter Property="Text" Value="Empty" />
    <!-- a felirata Empty lesz -->
```
- eseményre (*event trigger*), amelynek hatására valamilyen akciót (*trigger action*) hajtunk végre, pl.:

```
<EventTrigger Event="TextChanged">
    <local:NumberValidateAction />
</EventTrigger>
```
- a triggerek kombinálhatóak (*multi trigger*)

MAUI alapismeretek

Triggerek

- a trigger akciók (**TriggerAction<>**) olyan osztályok, amelyben az eseményre adott reakciót (**Invoke**) írhatjuk le, pl.:

```
public class NumberValidateAction
    : TriggerAction<Entry>
    // megadjuk az érintett vezérlő típusát
{
    protected override void Invoke (Entry entry) {
        // megadjuk a tevékenységet
        double result;
        entry.TextColor =
            Double.TryParse(entry.Text, out result)
            ? Color.Default : Color.Red;
        // ha nem szám, piros lesz a betűszín
    }
    ...
}
```

MAUI alapismeretek

Példa

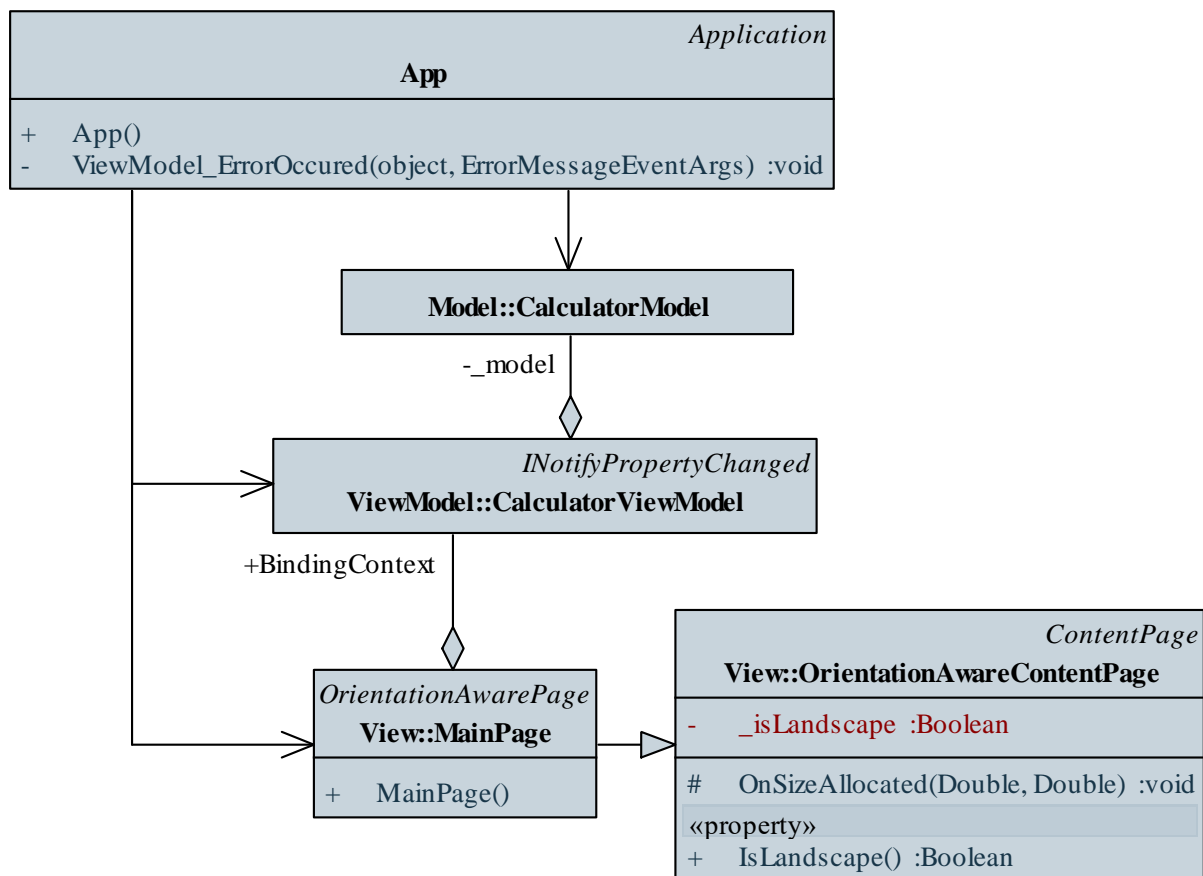
Feladat: Készítsünk egy egyszerű számológépet, amellyel a négy alapműveletet végezhetjük el, illetve láthatjuk korábbi műveleteinket is.

- 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**)
- megvalósítunk egy új laptípust (**OrientationAwareContentPage**), amely kezeli a tájolást (**IsLandscape**), így triggerek segítségével tudunk reagálni az elforgatásokra
- külön méreteket definiálunk mobilra és táblagépekre (**OnIdiom**)

MAUI alapismeretek

Példa

Tervezés:



MAUI alapismeretek

Példa

Megvalósítás (MainPage.xaml):

```
<view:OrientationAwareContentPage ...>
```

...

```
<Style x:Key="BasicButtonStyle"
```

```
  TargetType="Button">
```

```
  <Setter Property="FontSize">
```

```
    <Setter.Value>
```

```
      <OnIdiom x:TypeArguments="x:Double"
```

```
        Phone="15" Default="45" />
```

```
      <!-- reagálunk az eszközre -->
```

```
    </Setter.Value>
```

```
  </Setter>
```

...

MAUI alapismeretek

Példa

Megvalósítás (MainPage.xaml):

```
<view:OrientationAwarePage ...>
...
<StackLayout>
  <StackLayout.Triggers>
    <!-- reagálunk az elforgatásokra -->
    <DataTrigger TargetType="StackLayout"
      Binding="{Binding Source={x:Reference
        ContentPage}, Path=IsLandscape}"
      Value="True">
      <Setter Property="Orientation"
        Value="Horizontal" />
    ...
  </view:OrientationAwarePage>
```


MAUI alapismeretek

Időzítés

- Felületi időzítésre használhatjuk a platformfüggetlen időzítőt (**Device.StartTimer**)
 - megadhatjuk az időintervallumot, és a végrehajtandó tevékenység lambda-kifejezését
 - a tevékenység egy logikai függvény, amely amennyiben hamissal tér vissza, az időzítő leáll, pl.:

```
elapsedTime = 0;
```

```
Device.StartTimer(TimeSpan.FromSeconds(1), () =>  
{  
    entry.Text = ++elapsedTime;  
    // eltelt másodpercek kiírása  
    return elapsedTime < 120;  
    // 2 percig fut az időzítő  
});
```

MAUI alapismeretek

Időzítés

- Üzleti logikában történő időzítésre használható a párhuzamosított időzítő (**System.Threading.Timer**)
 - példányosításkor megadhatjuk a tevékenységet, egy felügyelt állapotot, a kezdési időt és az intervallumot, pl.:

```
elapsedTime = 0;  
timer = new Timer(state =>  
    { entry.Text = ++elapsedTime; },  
    state, 0, 1000); // 1 másodpercenként fut le
```
 - megsemmisítéssel (**Dispose**) leállíthatjuk az időzítőt
 - az időzítőt nem kell szinkronizálni a felülettel
- Amennyiben eseményvezérelt időzítőre van szükségünk, a funkcionalitást becsomagolhatjuk egy annak megfelelő típusba.
 - Valójában a **System.Timers.Timer** is ezt teszi.

MAUI alapismeretek

Példa

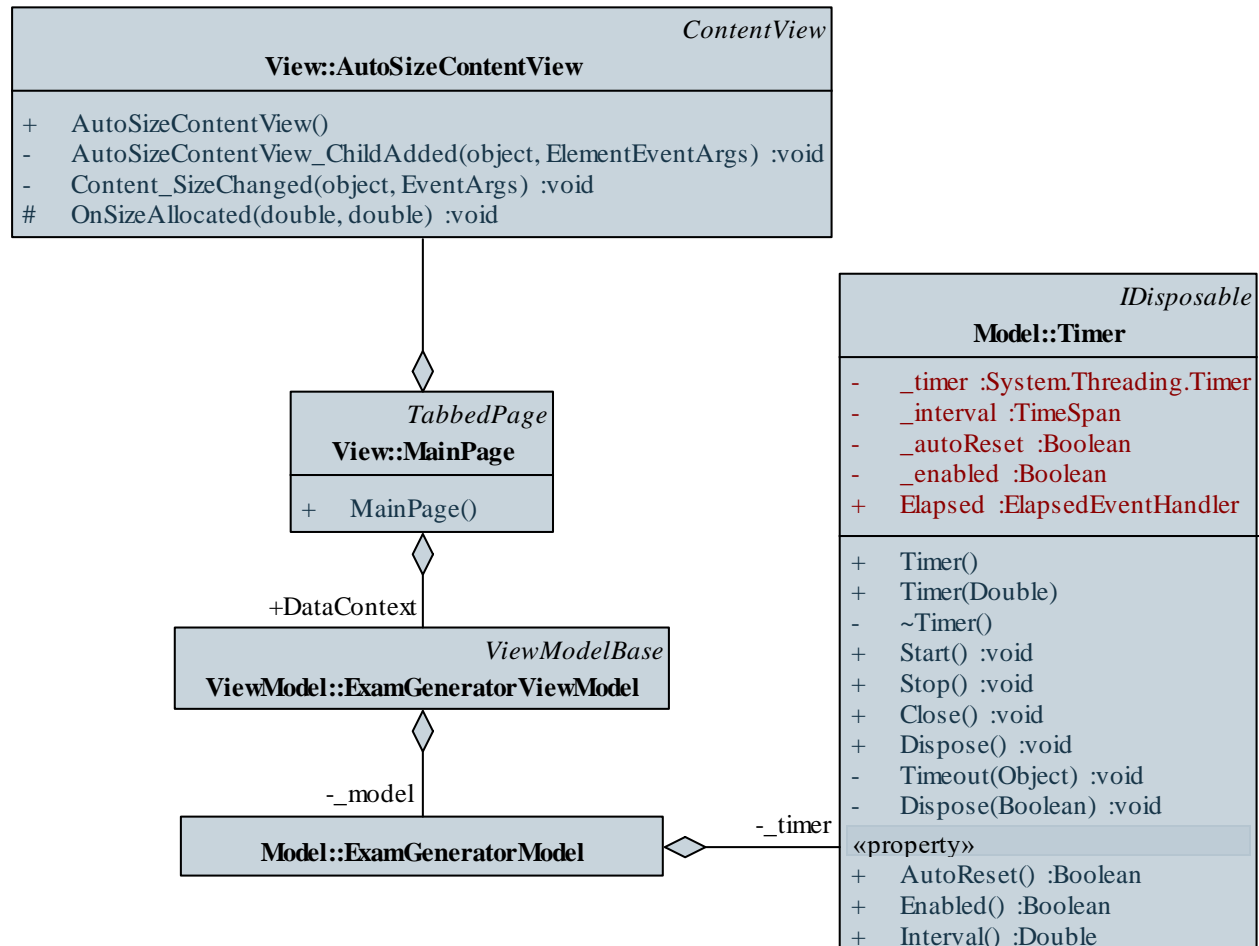
Feladat: Készítsünk egy vizsgatétel generáló alkalmazást, amely ügyel arra, hogy a vizsgázók közül ketten ne kapják ugyanazt a tételt.

- a megjelenítéshez két lapot (**ContentPage**) használunk, amelyek között lapozunk (**AppShell** használatával)
- az első lapon csak a tétel sorszámát jelenítjük meg, és érintéssel tudjuk generálni a sorszámot (**TapGestureRecognizer**), a második lapon elhelyezzük a beállításokat
- a sorszám méretét automatikusan méretezzük, ehhez egy saját vezérlőt veszünk fel (**AutoSizeContentView**), amely méretezi a tartalmát
- megvalósítunk egy saját időzítőt (**Timer**), amely teljes egészében megfelel az üzleti logikában használt időzítőnek (az időzített eseményt már nem kell szinkronizálnunk a nézetmodellben)

MAUI alapismeretek

Példa

Tervezés:



MAUI alapismeretek

Példa

Megvalósítás (MainPage.xaml):

```
<TabbedPage ...>
```

...

```
<view:AutoSizeContentView>
```

```
<!-- automatikusan méreteződik a tartalom -->
```

```
<Label Text="{Binding QuestionNumber}" ...>
```

```
<Label.GestureRecognizers>
```

```
<!-- érintés kezelése -->
```

```
<TapGestureRecognizer
```

```
  Command="{Binding StartStopCommand}" />
```

```
<!-- érintés -->
```

```
</Label.GestureRecognizers>
```

```
</Label>
```

```
</view:AutoSizeContentView >
```

...

MAUI alapismeretek

Példa

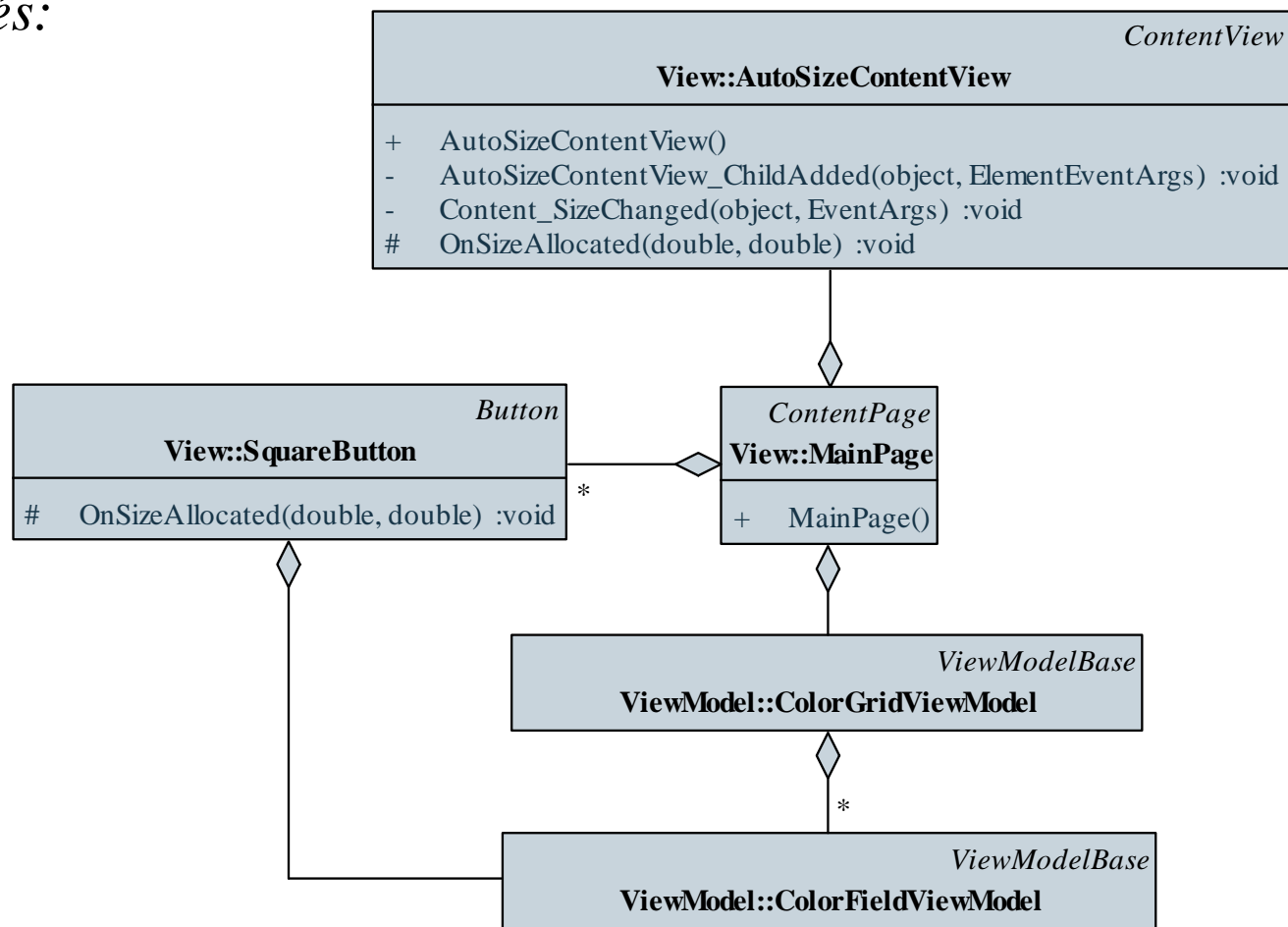
Feladat: Készítsünk egy dinamikus méretezhető táblát, amely három szín között (piros, fehér, zöld) állítja a kattintott gombot, valamint a vele egy sorban és oszlopban lévőket.

- felhasználjuk a rendelkezésre álló a nézetmodell (`ColorGridViewModel`, `ColorFieldViewModel`), a színváltást trigger segítségével vezéreljük
- használjuk a `Grid` elrendező vezérlőt a rács megjelenítéséhez
 - a rács dinamikus méretezéséhez a nézetmodellben (`ColorGridViewModel`) előállítjuk a megfelelő `RowDefinitionCollection` és `ColumnDefinitionCollection` objektumokat
 - ezekre már adatkötést végezhetünk a nézetben

MAUI alapismeretek

Példa

Tervezés:



MAUI alapismeretek

Példa

Megvalósítás (MainPage.xaml):

```
<Grid BindableLayout.ItemsSource="{Binding Fields}"
      RowDefinitions="{Binding RowDefinitions}"
      ColumnDefinitions="{Binding ColumnDefinitions}">
  <!-- gombrács adatkötéssel megadott
        dinamikus méretezéssel -->
  <BindableLayout.ItemTemplate>
    <DataTemplate>
      <Button ... />
      <!-- a rácsot gombokkal töltjük fel -->
      ...
    </DataTemplate>
  </BindableLayout.ItemTemplate>
</Grid>
```