Eseményvezérelt alkalmazások: 10. gyakorlat

A munkafüzet az *Avalonia UI keretrendszer* használatába vezet be minket. Az Avalonia UI egy a .NET környezetre épülő, cross-platform keretrendszer eseményvezérelt grafikus alkalmazások fejlesztésére. Támogat több asztali (Windows, Linux, macOS) és mobil (Android, iOS) platformot is, valamint WebAssemblyként böngészőben is futtatható. Lehetővé teszi a fejlesztést MV architektúrában is, de kifejezetten az MVVM architektúra használata javasolt.

A feladat egy több platformon is futtatható **aszinkron kép letöltő alkalmazás elkészítése**, amely segítségével egyszerűen listázhatjuk az egy weboldalon megjelenő képeket.

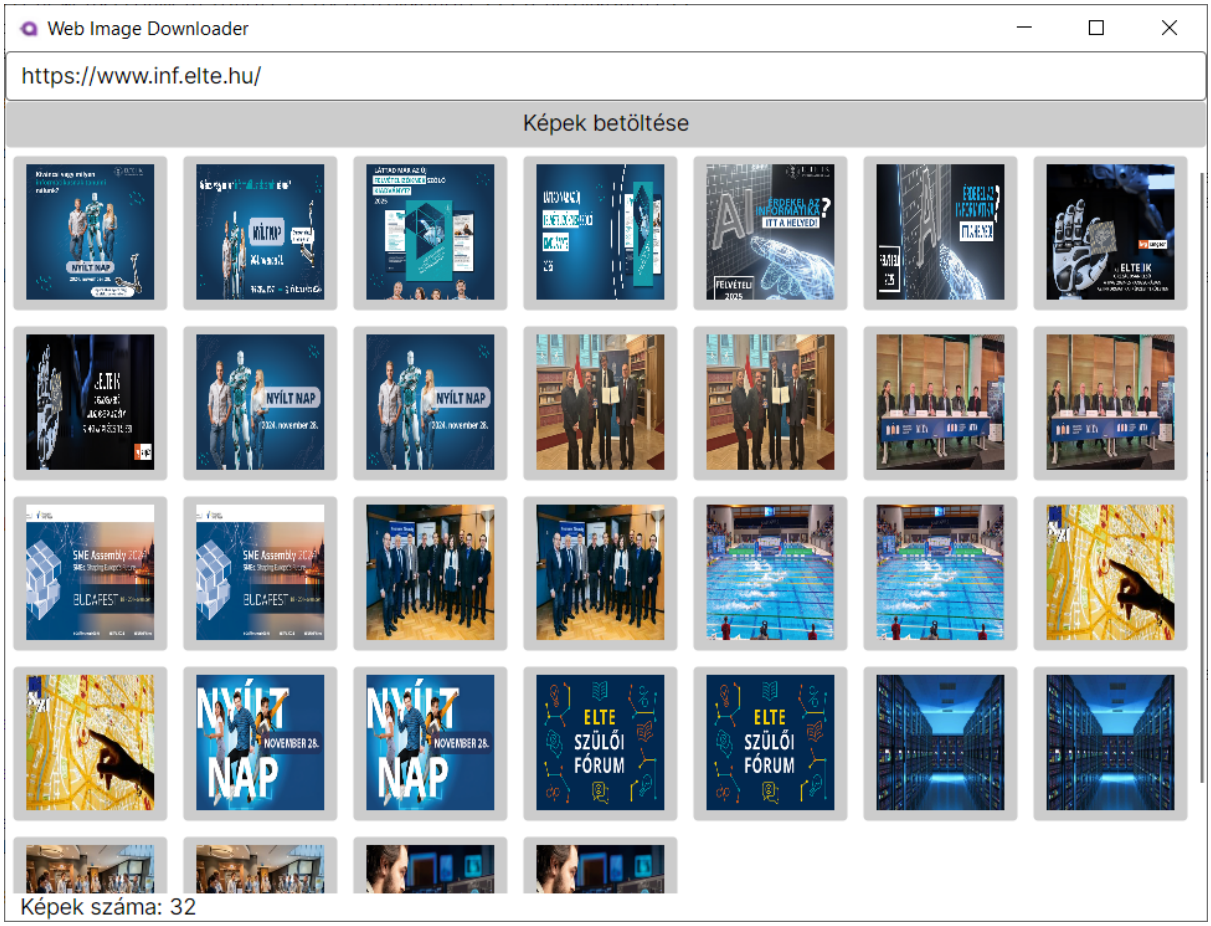

Figure 1: Windows App

Figure 2: Android App

Az alkalmazást *Avalonia UI* keretrendszerrel, háromrétegű (modell-nézet-nézetmodell) architektúrában, eseményvezérelt paradigma alapján valósítjuk meg. Az Avalonia UI használatával egyetlen kódbázist készítve fejleszthetünk olyan alkalmazást, amelyet több platformon is tudunk futtatni.

1 Előkészületek Avalonia UI alkalmazás fejlesztéséhez ^{KM}

Első lépésként telepítenünk kell az *Avalonia for Visual Studio 2022* kiegészítőt a Visual Studio-hoz.

1. Indítsuk el a Visual Studio-t és válasszuk az *Extensions* menüből a *Manage extensions* menüpontot.
2. Keressünk rá az *Avalonia* kiegészítőre, majd telepítsük. Ezzel elérhetővé válik az Avalonia UI felületű alkalmazások grafikus tervezése és debuggolása Visual Studio-ban.
3. Telepítsük az *Avalonia Template Studio* kiegészítőt is, ezzel új projekt létrehozásakor olyan Avalonia UI-os sablonok közül is választhatunk, amelyek egy összetettebb varázslóval vezetnek végig a folyamaton.

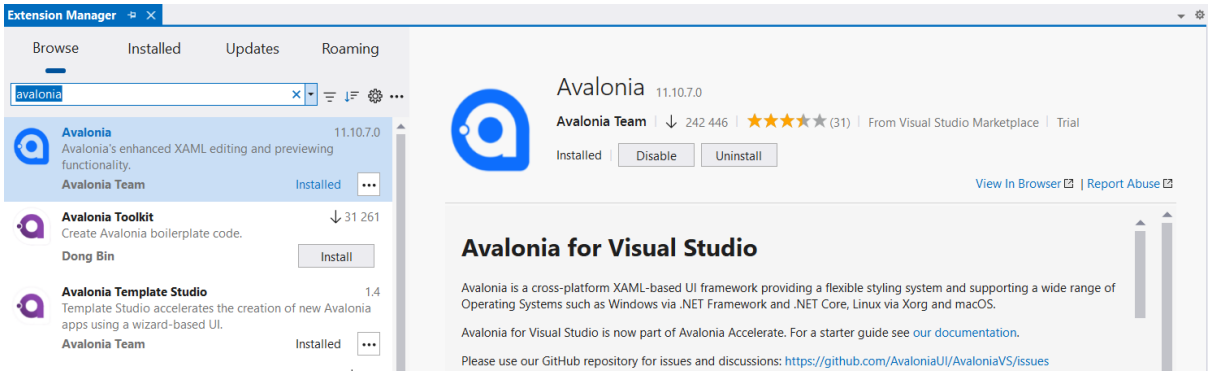

Figure 3: Kiegészítők kezelése Visual Studio-ban

Amíg az Avalonia UI open-source és ingyenes, addig a VS kiegészítő az *Avalonia Accelerate* része, amely profitorientált felhasználásra már nem feltétlen ingyenes. Legkésőbb 21 hét után szükséges Avalonia fejlesztői fiókot regisztrálunk, de a kiegészítő közösségi verziója továbbra is ingyenes - amennyiben vállalatunk éves bevétele az 1 millió USD-t nem éri el.

JetBrains Rider fejlesztőkörnyezet használata esetén az *AvaloniaRider* plugin telepítésével kaphatunk hasonló támogatást az IDE-ben. A projekt sablonokat az alábbi konzol utasítással telepíthetjük:
`dotnet new install Avalonia.Templates`

1.1 Android platform fejlesztési támogatása

Második lépésként, az Android platformra fejlesztéshez telepítenünk kell a Java SDK-t, az Android SDK-t és a .NET SDK Android Workload-ot. A legegyszerűbben ezt úgy tehetjük meg, ha a Visual Studio 2022-höz a teljes *Multi-platform App UI (MAUI) development* csomagot telepítjük. (A géptermi gépeken ez már előzetesen telepítésre kerül.)

1. A telepített programok közül válasszuk ki a *"Visual Studio Installer"* alkalmazást.
2. A megjelenő listából válasszuk ki a Visual Studio 2022-t, majd kattintsunk a *Modify* gombra.
3. A Workloads fülön pipáljuk be a *.NET Multi-platform App UI development* lehetőséget, majd kattintsunk a *Modify* gombra.

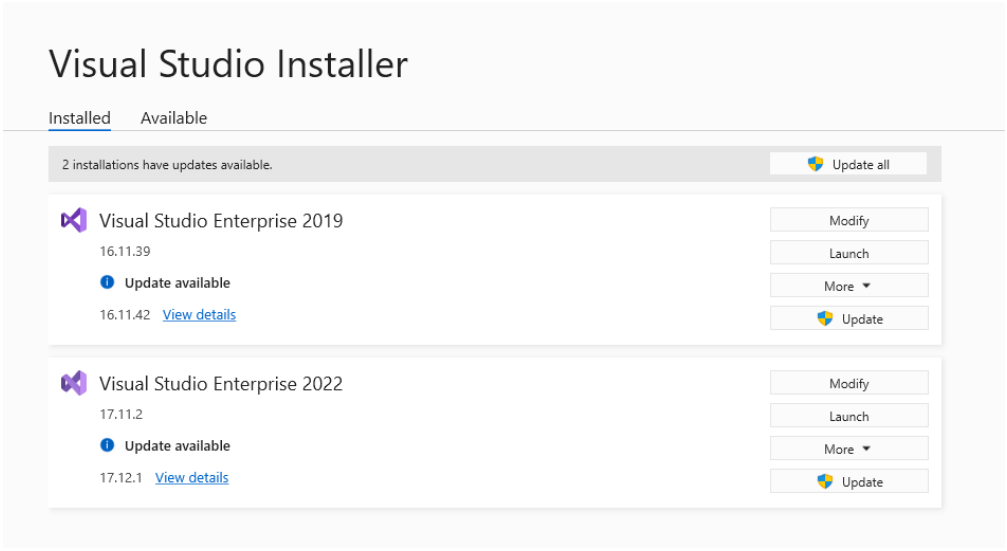

Figure 4: Installer lista

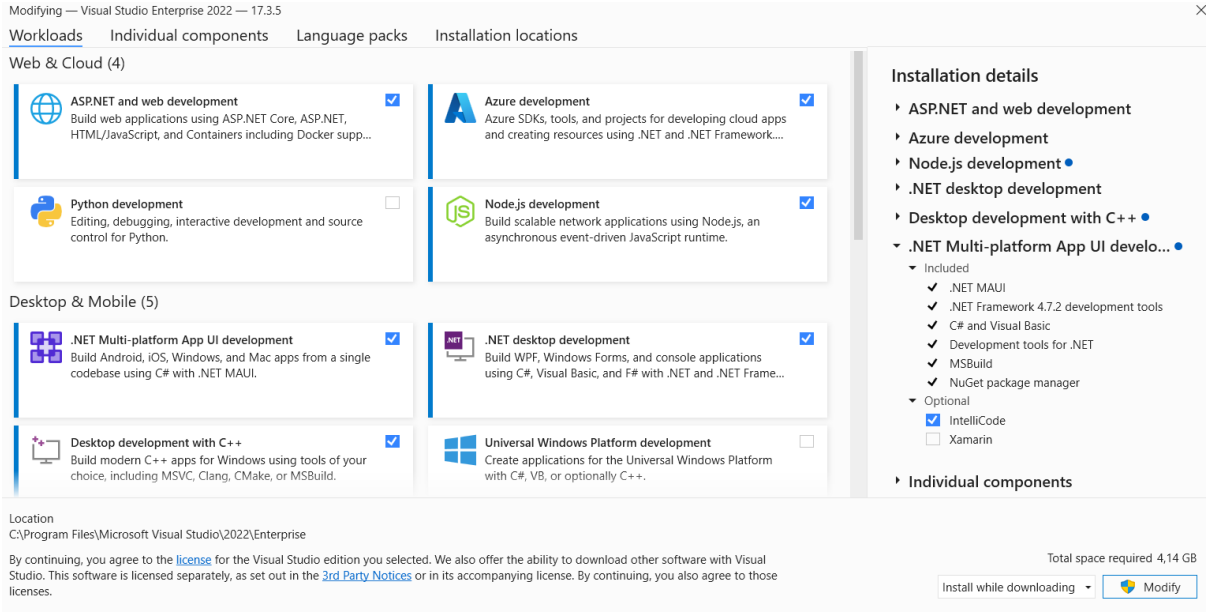

Figure 5: Installer Workloads: MAUI

> *JetBrains Rider* fejlesztőkörnyezet használata esetén külön szükséges Java SDK-t és Android SDK-t telepítenünk egy preferált forrásból. A .NET SDK Android Workloadja a következő utasítással telepíthető:
> `dotnet workload install android`

1.2 Hyper-V engedélyezése

A Visual Studioval egyszerűen futtathatóak, tesztelhetőek és debuggolhatóak az Avalonia UI alkalmazások virtuális Androidhoz eszközön is, emulátor segítségével. Ha azonban a hardveres gyorsítás (*hardware acceleration*) nem érhető el az adott számítógépen (vagy nincs engedélyezve), akkor az emulátor jellemzően meglehetősen lassú lesz. A hardveres gyorsítás engedélyezésével az emulálás sebessége szignifikánsan javítható.

A hardveres gyorsítást legegyszerűbben a Hyper-V engedélyezésével kapcsolhatjuk be, ehhez rendszergazdaként a *Windows-szolgáltatások be-és kikapcsolása* vezérlőpanelen (angol nyelvű operációs rendszer esetén: *Turn Windows features on or off*) kapcsoljuk be a jelölt Hyper-V szolgáltatásokat, majd indítsuk újra a számítógépet.

A géptermi gépeken a Hyper-V már engedélyezve van!

> *Tipp:* A Hyper-V a Windows 10/11 *Home* verziójában nem érhető el, azonban az Azure Dev Tools for Teaching akadémiai együttműködési program keretében a Visual Studio 2022 Enterprise verziója mellett többek között a Windows 10/11 Education verziója is jogtisztán beszerezhető az ELTE IK hallgatói számára (INF-es azonosítóval bejelentkezve). A Windows 10/11 Education verziója a Pro változattal közel azonos funkciókészlettel rendelkezik.
> Amennyiben valaki már telepített Windows 10/11 Home verzióval rendelkezik, frissíthet is az Education verzióra. Ehhez a *Gépház -> Névjegy* ablakban (angol változatban: *Settings -> About*) van lehetőség, az *Azure Dev Tools for Teaching*-ből beszerzett termékkulcsot megadva.

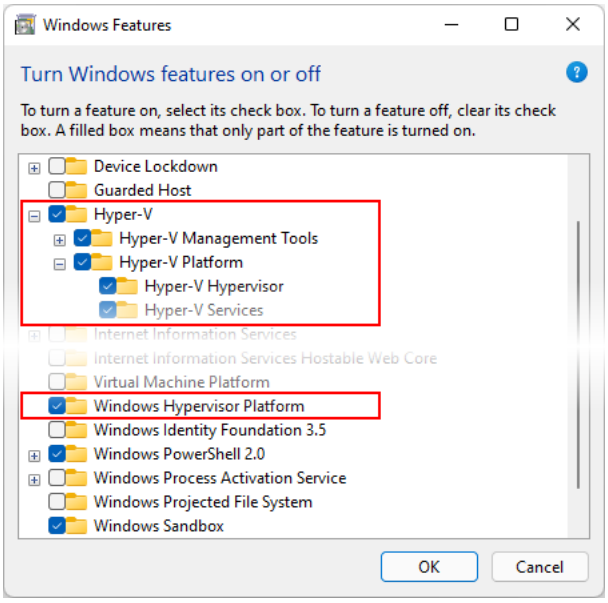

Figure 6: Hyper-V szolgáltatások bekapcsolása

1.3 Fizikai Android eszköz használata ^{OP}

Alternatívaként amennyiben rendelkezünk vele, fizikai Android eszközünkre is fejleszthetünk, az alkalmazást azon is tesztelhetjük, debuggolhatjuk. A MAUI-hoz elérhető dokumentáció Avalonia esetén is aktuális, hogyan tudunk egy fizikai eszközt könnyedén Visual Studiohoz kapcsolni vezetékesen (USB kábelen keresztül) vagy vezeték nélkül (WiFi).

Fontos, hogy előtte a fejlesztői módot az eszközön engedélyezni kell, valamint figyeljük a telefon (vagy tablet) kijelzőjét is, mert az első csatlakoztatáskor jóvá kell hagynunk a Visual Studio kapcsolódását.

1.4 iOS platform fejlesztési támogatása ^{OP}

Technikai akadálya nem lenne, de Windows operációs rendszerről licenszelési problémák miatt nem tudunk sem iOS-re se macOS-re fordítani. Ehhez szükségünk lesz egy XCode-ot futtató Mac-re, fizikai eszköz vagy emulátor (szimulátor) használat esetén is. Amennyiben valaki rendelkezik ilyen eszközökkel, innentől alapvetően két lehetősége van:

- Avalonia alkalmazásunkat Mac-en fejlesztjük (van Visual Studio Mac-re is, Azure Dev Tools for Teachingből letölthető) és ott az alkalmazást iOS Simulátorban futtatjuk vagy telepíthetjük egy kapcsolt fizikai eszközre is. XCode telepítésére szükség lesz. Ld. a dokumentációt.
- Egy Windowsos fejlesztői számítógépen dolgozunk továbbra is, ami egy macOS-es számítógépen keresztül telepíti az alkalmazást a telefonra. Ez már kicsit összetettebb lépéseket igényel, konfigurálása egyszeri 20-30 percet igényelhet. Az ehhez vonatkozó MAUI-s dokumentáció Avalonia esetében is használható. Megjegyzések:
  - XCode telepítése itt is szükséges lesz a macOS-es gépre. Fontos, hogy legalább egyszer el is kell indítani, mert az első indítással lesz teljes a telepítés.
  - Ha az XCode már telepítve van, de mégis olyan hibaüzenet jelentkezik, mintha nem lenne, érdemes megpróbálni azt, hogy az XCode-on belül a *Locations* menü *Command line tools* beállításánál újra kiválasztani az egyébként is kiválasztott elemet a listából.

2 Projekt létrehozása ^{KM}

Készítsünk Visual Studioban egy új *Avalonia C# Project*-et, a *solution* és a *projekt* neve legyen *ImageDownloader.Avalonia*.

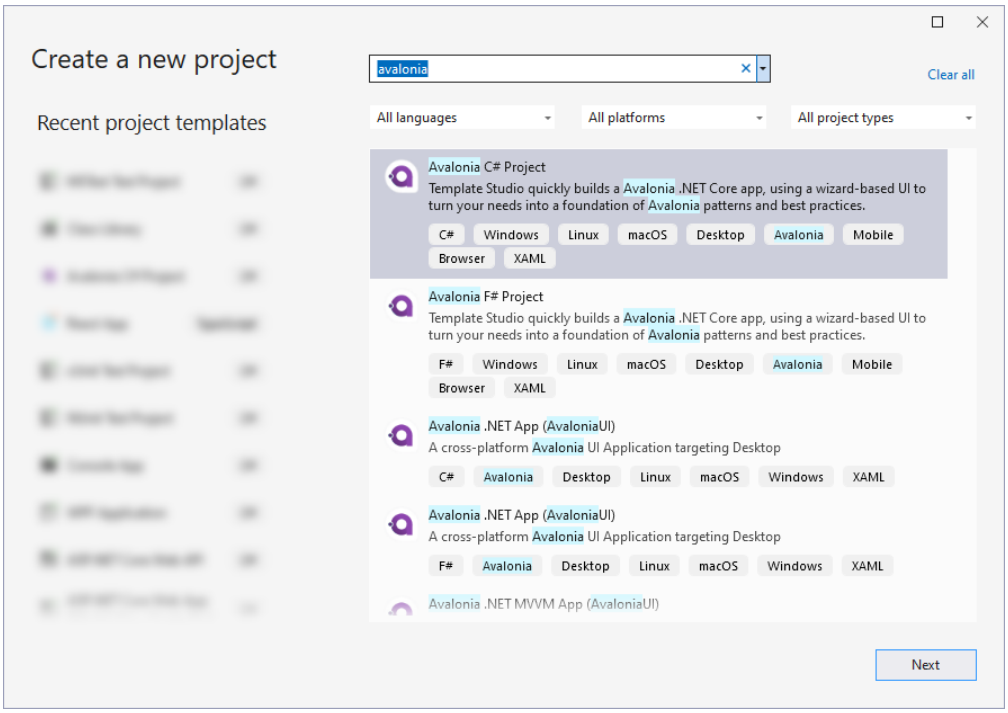

Figure 7: Projekt létrehozása

A projekt létrehozását egy több lépéses varázsló segíti. Válasszuk ki a *Desktop* és *Android* platformokat támogatásra, alkalmazott tervezési mintaként (*design pattern*) pedig a *Community Toolkit*-et. A varázsló 3. lépésben a *Feature*-k kiválasztása most nem fontos.

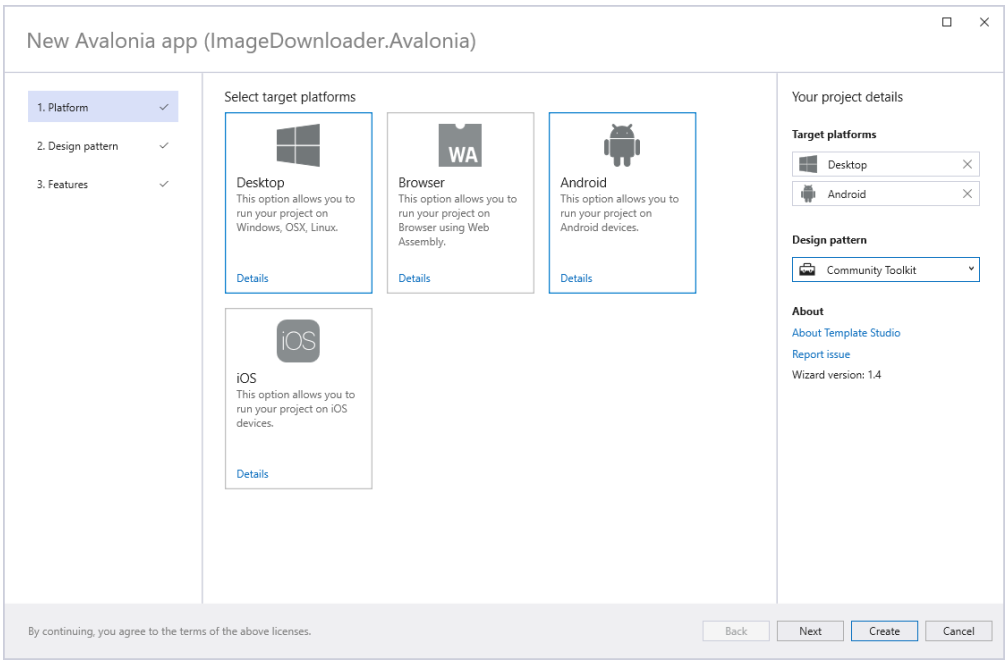

Figure 8: Támogatott platformok és tervezési minta kiválasztása

3 Fordítás több platformra ^{KM}

Az Avalonia alkalmazások multi-projekt felépítést követnek, azaz a `ImageDownloader.Avalonia` projekt egy *Class Library* projekt, és a `.Desktop`, valamint `.Android` projektekből készülnek futtatható binárisok. A futtatás az egyes platformokon a következő módon lehetséges:

- Windows-on: válasszuk ki a `ImageDownloader.Avalonia.Desktop` projektet, és a megszokott módon fordítsuk, illetve futassuk. A programunk így fordított változata a többi támogatott asztali operációs rendszeren is futtatható.
- Androidon: válasszuk ki a `ImageDownloader.Avalonia.Android` projektet, majd mellette egy fizikai eszközt vagy emulátort a futtatáshoz.
  - Ahhoz, hogy egy fizikai Android készülékre tudjunk fordítani, a készüléknek fejlesztői módban kell lennie, illetve a fejlesztői beállításoknál az USB hibakeresésnek engedélyezve kell lennie. Ld. a kapcsolódó MAUI-s dokumentációt ennek beállításához.
  - Emulátort a *Tools -> Android -> Android Device Manager* menüpontban hozhatunk létre, ha alapértelmezetten nem jött volna létre egy a MAUI workload telepítésekor.

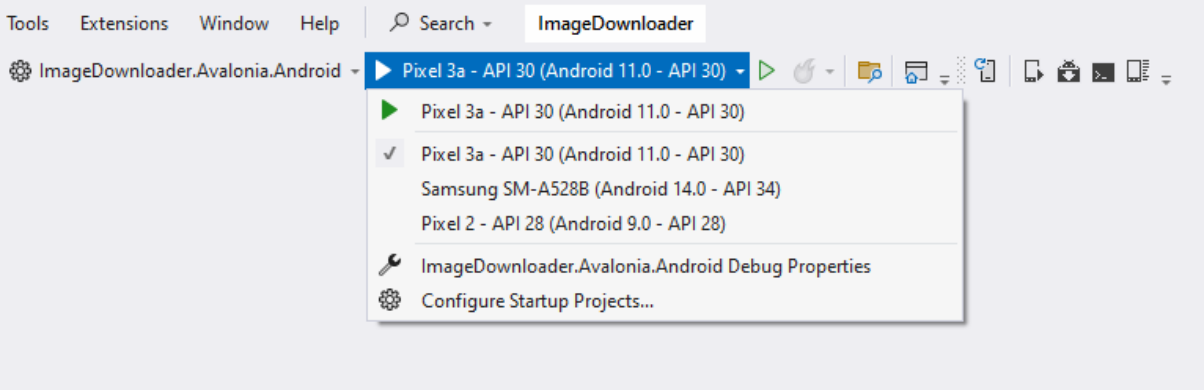

Figure 9: Alkalmazás indítása Androidon

4 Modell réteg megvalósítása ^{KM}

A modell réteget nem szükséges megvalósítanunk, hiszen az alkalmazás a modell réteg tekintetében megegyezik a *8. gyakorlaton* megvalósított képletöltő alkalmazásunk modell rétegével.

A 8. gyakorlaton létrehozott WPF alkalmazás egyetlen projektként készült el, azaz nincs szétbontva külön model és view projektre. A teljes projekt pedig a WPF mivolta miatt csak a Windows platformot támogatja (a jelenlegi alkalmazásunk pedig cross-platform).

Megjegyzés: ez jól látható a *csproj* fájlokat megnyitva a `TargetFramework` mezőben is.
WPF-es ImageDownloader:

```
<TargetFramework>net8.0-windows</TargetFramework>
```

Avalonia UI-os ImageDownloader (platformfüggetlen projekt):

```
<TargetFramework>net8.0</TargetFramework>
```

Emiatt a korábbi *ImageDownloader* projekt model részét emeljük át a jelenlegi solution-be egy új, platformfüggetlen *Class Library* projektbe. Ezen *class library* projekt neve lehet például: *ImageDownloader.Model*. A keretrendszernél itt választhatjuk a .NET 8-at, mint *LTS* verziót.

Adjuk hozzá az *ImageDownloader.Avalonia* projektünkhöz a létrehozott *ImageDownloader.Model* projektet függőségként (*"Add Project Reference"*).

5 Nézetmodell réteg megvalósítása ^{KM}

Az *ImageDownloader.Avalonia* projektben már adott egy *ViewModels* mappa. A benne lévő fájlokat töröljük, és helyette az alábbi osztályokat / fájlokat hozzuk létre:

- `DelegateCommand`: megvalósítása egyezzen meg a *8. gyakorlaton* használt megvalósítással.
- `ViewModelBase`: megvalósítása egyezzen meg a *8. gyakorlaton* használt megvalósítással.
- `MainViewModel`: megvalósítását lásd lejjebb.

5.1 MainViewModel osztály

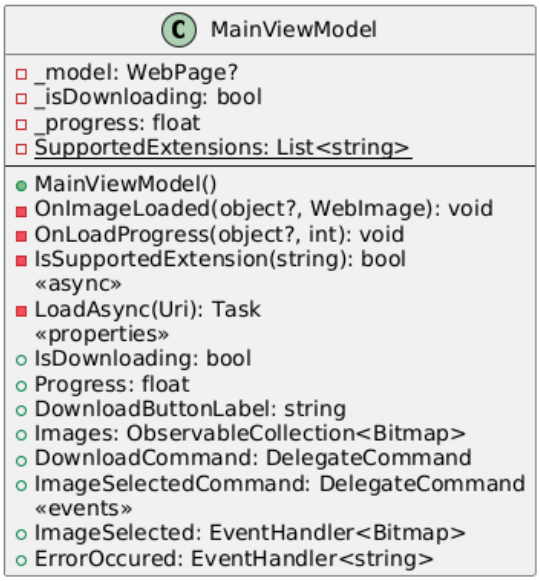

Figure 10: `MainViewModel` osztálydiagramja

A `MainViewModel` nagyrészt megegyezik a *8. gyakorlaton* elkészített `MainViewModel` osztállyal, viszont szükséges néhány átalakítás. Ezért induljunk ki ebből.

5.2 Képek kezelése

Avalonia UI-ban a WPF-es `System.Windows.Media.Imaging.BitmapImage` típus helyett az `Avalonia.Media.Imaging.Bitmap` típust használhatjuk az alábbi helyeken:

- `Images` tulajdonság,
- `ImageSelected` esemény,
- a konstruktorban az `Images` példányosításakor,
- az `ImageSelectedCommand`-ban az argumentum típsuának ellenőrzésekor.

5.3 `LoadAsync` átalakítása

Kezeljük a `_model.LoadImagesAsync()` hívás által dobott esetleges kivételeket, hiszen mobil eszköz esetén a Internet kapcsolat stabilitása is kevésbé megbízható. Avalonia UI keretrendszerben üzenetablak feldobásához használjuk a `MessageBox.Avalonia` NuGet csomagot. Az üzenetablakot azonban ne a nézetmodell jelenítse meg, hanem definiáljunk egy eseményt (pl. `ErrorOccured`), amelyre a vezérlési rétegben feliratkozva tudjuk majd megjeleníteni a felhasználónak a hibaüzenetet.

A `MessageBox.Avalonia` NuGet csomag telepítése előtt érdemes lehet a projekt sablon által hozzáadott Avalonia NuGet csomagok frissítése, az esetleges kompatibilitási problémák elkerülése érdekében.

```
await MessageBoxManager.GetMessageBoxStandard(
    "Cím",
    "Üzenet",
    ButtonEnum.Ok, // gombok
    Icon.Error)    // ikon
.ShowAsync();
```

5.4 `OnImageLoaded` eseménykezelő átalakítása

Az `ImageLoaded` esemény minden kiváltásakor a modell sikeresen letöltött egy új képet, ennek megfelelően ezt a nézeten is megjeleníthetjük.

1. Ellenőrizzük, hogy a letöltött kép kiterjesztése az alábbiak közül valamelyik-e: *jpg, jpeg, png, gif*. Ezt az eseményargumentumban érkezett `webImage.Url.LocalPath` vizsgálatával tudjuk megtenni.
2. Készítsünk egy új memóriabeli képet: `new MemoryStream(webImage.Data)`
3. A `MemoryStream`-ből készítsünk egy `Bitmap`-et, amit már az `Images` kollekcióhoz adhatunk.

6 Nézet réteg megvalósítása ^{EM}

Az *ImageDownloader.Avalonia* projektben már adott egy *Views* mappa.

A fő nézetet (`MainView.axaml`) a *8. gyakorlaton* megvalósított WPF ablak mintájára készítsük el. Mindössze a következő eltérésekre kell készülnünk:

- Az URL címet bekérő szövegdoboz tartalmára a gomb parancsának paraméterénél eltérő, egyszerűbb szintaxissal tudunk hivatkozni:
  `{Binding #UrlTextBox.Text}`
- Nem lesz szükségünk a `BooleanToVisibilityConverter`-re, ugyanis az Avalonia UI-ban a `ProgressBar`-nak egy logikai értéket váró `IsVisible` tulajdonsága van.
- Nincsen `StatusBar` panel az Avalonia UI keretrendszerben, helyette egy horizontálisan orientált `StackPanel`-t használhatunk.
- A képeket az `Image` vezérlővel jeleníthetjük meg. (Majd a következő, *11. gyakorlaton* tesszük kattinthatóvá a képeket.)

7 Vezérlés ^{EM}

Kapcsoljuk hozzá a létrehozott nézetmodellünket a nézethez. Ezt az alábbi lépésekben tegyük meg:

1. Az `App.axaml.cs` osztály `OnFrameworkInitializationCompleted` metódusában hozzunk létre egy új `MainViewModel` példányt.
2. A `DataContext` *propertynek* állítsuk be a létrehozott `MainViewModel`-t, mind az asztali, mind a mobilos alkalmazás életciklus esetén.
3. Ne feledkezzünk meg feliratkozni a nézetmodell `ErrorOccured` eseményére és jelenítsük meg a kért hibaüzenetet egy felugró ablakban a `MessageBoxManager` használatával.