Pannon Egyetem   
Műszaki Informatikai Kar  
Rendszer- és Számítástudományi Tanszék  
Programtervező és Mérnökinformatikus BSc

PROJEKT LABOR

Fotókönyv szerkesztő szoftver C# nyelven

Englert Gyula  
Mátics Dániel  
Bercza Ferenc Dániel

Témavezető: Dr. Heckl István

2024

**PROJEKTLABOR TÉMAKIÍRÁS**

Programtervező és Mérnökinformatikus BSc szakos hallgatók részére

**Fotókönyv szerkesztő szoftver C# nyelven**

Témavezető: Dr. Heckl István

**A feladat leírása:**

A tervezendő C# (WPF) grafikus alkalmazás egy fotókönyv szerkesztő program lesz. Ennek a programnak képesnek kell lennie keretek kezelésére, amelyekben a képeket mozgathatjuk és nagyíthatjuk. A keretek pozíciójának és méretének szabályozhatónak kell lennie, valamint a háttérkép színtónusának állítására is lehetőséget kell biztosítani. A programnak tartalmaznia kell különböző oldalelrendezési opciókat, varázsló funkciót a könnyebb használhatóság érdekében, szövegbeviteli lehetőségeket, valamint lehetőséget kell adnia képregény-stílusú szövegbuborékok hozzáadására a képekhez.

**Feladatkiírás:**

* Meglévő rendszerek tanulmányozása
* Felhasználói felület tervezése, létrehozása
* Képszerkesztési funkciók tervezése, létrehozása
* Fotókönyv állapotának mentése
* Elkészült fotókönyv mentése, nyomtatása
* Felhasználói élmény érdekében történő grafikai elemek hozzáadása

Dr. Heckl István Dr. Heckl István

Egyetemi docens Egyetemi docens

Témavezető Tárgyfelelős

Tartalmi összefoglaló

Célunk egy C# programozási nyelvben megvalósított, Windows Presentation Foundation (WPF) technológiára épülő asztali fotókönyvkészítő alkalmazás kifejlesztése volt. A projekt során kiemelt célunk volt egy olyan felhasználóbarát, intuitív és funkciógazdag szoftver létrehozása, amely a modern szoftverfejlesztési elveknek megfelelően épül fel.

Az alkalmazás architektúrájának megtervezésekor a Model-View-ViewModel (MVVM) tervezési minta és a Dependency Injection (DI) technika alkalmazása mellett döntöttünk. Az MVVM minta lehetővé tette az alkalmazás logikájának és felhasználói felületének szétválasztását, növelve ezzel a kód karbantarthatóságát és tesztelhetőségét. A DI pedig a komponensek közötti kapcsolatok rugalmas kezelését biztosította, elősegítve a moduláris felépítést.

A projekt eredménye egy jól strukturált, karbantartható és bővíthető C# WPF alkalmazás, amely megfelel a kitűzött céloknak. A fejlesztés során szerzett tapasztalatok alapján kijelenthető, hogy az MVVM és DI minták, valamint a GitHub használata kulcsfontosságú elemei a modern szoftverfejlesztési projekteknek.

**Kulcsszavak: C#, WPF, MVVM, Dependency Injection, GitHub, szoftverfejlesztés, fotókönyv készítő**

Abstract

Our goal was to develop a desktop photo book creation application implemented in the C# programming language and based on Windows Presentation Foundation (WPF) technology. Throughout the project, our primary objective was to create a user-friendly, intuitive, and feature-rich software that adheres to modern software development principles.

When designing the application's architecture, we decided to use the Model-View-ViewModel (MVVM) design pattern and Dependency Injection (DI) technique. The MVVM pattern allowed for the separation of the application's logic and user interface, thus enhancing code maintainability and testability. The DI facilitated flexible handling of component relationships, promoting a modular structure.

The project resulted in a well-structured, maintainable, and expandable C# WPF application that meets the set goals. Based on the experiences gained during development, it can be stated that the MVVM and DI patterns, along with the use of GitHub, are crucial elements of modern software development projects.

**Keywords: C#, WPF, MVVM, Dependency Injection, GitHub, software development, photo book creator**

Tartalomjegyzék

[1. Bevezetés 1](#_Toc183785344)

[2. Felhasznált technológiák 2](#_Toc183785345)

[2.1 A C# programozási nyelv 2](#_Toc183785346)

[2.2 Visual Studio 2022 3](#_Toc183785347)

[2.3 .NET 8.0 3](#_Toc183785348)

[2.4 WPF (Windows Presentation Foundation) 4](#_Toc183785349)

[2.5 Model–view–viewmodel 5](#_Toc183785350)

[2.6 Dependency injection 6](#_Toc183785351)

[2.7 GitHub 6](#_Toc183785352)

[3. Versenytársak elemzése 7](#_Toc183785353)

[3.1 Journi 7](#_Toc183785354)

[3.2 CEWE 8](#_Toc183785355)

[3.3 InternetPrint 9](#_Toc183785356)

[3.4 MyPrintPix 10](#_Toc183785357)

[3.5 Sooters 11](#_Toc183785358)

[3.6 Prodigi 11](#_Toc183785359)

[3.7 Canva 12](#_Toc183785360)

[3.8 Táblázatos összehasonlítás 14](#_Toc183785361)

[3.9 Összefoglalás 14](#_Toc183785362)

[4. Funkciók ismertetése 15](#_Toc183785363)

[4.1 Felhasználói felület 15](#_Toc183785364)

[4.1.1 Navigációs sáv 15](#_Toc183785365)

[4.1.1.1 A fájl menü 15](#_Toc183785366)

[4.1.1.2 A szerkesztés menü 17](#_Toc183785367)

[4.1.2 Bal oldali panel 18](#_Toc183785368)

[4.1.2.1 Varázsló 18](#_Toc183785369)

[4.1.2.2 Képek hozzáadása 20](#_Toc183785370)

[4.1.2.3 Képkeretek hozzáadása 20](#_Toc183785371)

[4.1.2.4 Szövegdobozok hozzáadása 20](#_Toc183785372)

[4.1.2.5 Szövegbuborékok hozzáadása 21](#_Toc183785373)

[4.1.2.6 Sablon oldalak hozzáadása 21](#_Toc183785374)

[4.1.2.7 Színválasztó 21](#_Toc183785375)

[4.1.3 Fő szerkesztő felület 22](#_Toc183785376)

[4.1.4 Jobb oldali kontextusfüggő panel 25](#_Toc183785377)

[5. Megvalósítás 27](#_Toc183785378)

[5.1 App.xaml.cs 27](#_Toc183785379)

[5.2 MainWindow.xaml 27](#_Toc183785380)

[5.3 Tárolók 28](#_Toc183785381)

[5.3.1 NavigationStore 28](#_Toc183785382)

[5.3.2 ModalNavigationStore 29](#_Toc183785383)

[5.3.3 SelectedElementStore 29](#_Toc183785384)

[5.3.4 ProjectStore 30](#_Toc183785385)

[5.4 ResizeAdorner 38](#_Toc183785386)

[5.5 FrameAdorner 39](#_Toc183785387)

[5.6 Project.cs 40](#_Toc183785388)

[5.7 Főbb parancsok 41](#_Toc183785389)

[5.7.1 Varázsló 42](#_Toc183785390)

[5.7.2 Színválasztó 43](#_Toc183785391)

[5.7.3 Képkeretek, szövegbuborékok 44](#_Toc183785392)

[5.7.4 Sablonok 45](#_Toc183785393)

[5.7.5 Kijelölt oldal törlése 46](#_Toc183785394)

[5.7.6 Kijelölt oldal elforgatása 48](#_Toc183785395)

[5.7.7 Kijelölt oldal mentése sablonként 49](#_Toc183785396)

[5.8 Képszerkesztési műveletek 50](#_Toc183785397)

[5.8.1 Elemek forgatása 50](#_Toc183785398)

[5.8.2 Elemek tükrözése 51](#_Toc183785399)

[5.8.3 Elemek pozícionálása 52](#_Toc183785400)

[5.8.4 Elemek méretezése 53](#_Toc183785401)

[5.8.5 Rétegek kezelése 53](#_Toc183785402)

[5.8.6 Szöveges mezők betűméretének, betűtípusának és betűstílusának kezelése 54](#_Toc183785403)

[6. Összegzés 56](#_Toc183785404)

[Irodalomjegyzék 57](#_Toc183785405)

# 1. Bevezetés

A vizuális kommunikáció korában a fotókönyvek egyre népszerűbbé válnak, mint személyes emlékek őrzői és ajándékok. A piacon elérhető számos fotókönyv-készítő szoftver ellenére még mindig van tér egy olyan innovatív asztali alkalmazás számára, amely a felhasználók igényeit magasabb szinten elégíti ki.

A projektmunka fő célja egy fotókönyvek elkészítésére alkalmas asztali alkalmazás fejlesztése. Manapság számos hasonló webes, illetve asztali alkalmazás áll rendelkezésre a piacon, így fontos, hogy valamilyen módon megkülönböztessük a szoftverünket, és egyfajta plusz funkcionalitást biztosítsunk, amitől az általunk fejlesztett alkalmazás jobbá válik, mint a versenytársak, és ezáltal megérné minket választani a potenciális felhasználóknak. Egy ennyire telített piacon ez nagy nehézséget tud jelenteni, ellenben segítségünkre is lehet, hiszen rengeteg inspirációt tudunk nyerni ezekből a hasonló szoftverekből, megoldásokból.

Az egyik legfontosabb szempont az, hogy könnyen használható legyen a szoftverünk, és átlátható, letisztult felhasználói felülettel rendelkezzen. Ezen felül egy másik fontos szempont a varázsló funkció, mely nagyban megkönnyíti az alkalmazás használatát, és a kevésbé hozzáértő felhasználóknak is lehetővé teszi, hogy szép, esztétikus fotókönyveket készítsenek, könnyedén és rövid idő alatt. Ezekről a funkciókról, más alkalmazásokról és az összehasonlításokról a későbbiekben bővebben írunk.

Célkitűzésünk, hogy kifejlesszünk egy olyan asztali alkalmazást, amely megkülönbözteti magát vizuális megjelenésével, átlátható, egyszerű kezelőfelületével, és számos funkciójával új dimenziókat nyit a fotókönyvek készítésében. A szoftverünk segítségével a felhasználók könnyedén és gyorsan igényeiknek megfelelő fotókönyveket alkothatnak.

Az alkalmazás fejlesztése során kiemelt figyelmet fordítunk a felhasználói élményre. A könnyen kezelhető felület, a részletes funkciók, és a gyors válaszidő biztosítják, hogy a felhasználók zökkenőmentesen navigálhassanak a program funkciói között.

Az alkalmazásunkban a felhasználók számos testreszabási lehetőséggel rendelkeznek. A széleskörű tervezési eszközök segítségével a felhasználók szabadon alakíthatják fotókönyveiket, és olyan egyedi alkotásokat hozhatnak létre, amelyek tükrözik személyiségüket és stílusukat.

# 2. Felhasznált technológiák

## 2.1 A C# programozási nyelv

A programozási nyelv kiválasztásánál számos szempontot figyelembe kellett vennünk. Elsősorban az volt a legfontosabb, hogy a csoport tagjainak mely programozási nyelvekben van elegendő tapasztalata egy ilyen, nagyobb hangvételű projekt megvalósításához. Ezek közül pedig ki kellett választanunk azt, amely leginkább illeszkedik a problémánkra, mind elérhető eszközök, funkciók, fejlesztői környezetek, keretrendszerek stb. tekintetében. Ezáltal esett a választás a C# programozási nyelvre. Mindhármunk rendelkezik megfelelő tapasztalattal ebben a nyelvben, és eszköztára lehetővé teszi, hogy hatékonyan megvalósítsuk benne a feladatot.

A C# programnyelv egy, a Microsoft által fejlesztett, 2001-ben megjelent programozási nyelv. A .NET keretrendszer részeként került fejlesztésre. Nagy mértékben eltér a C és C++ nyelvektől, azoknak egyfajta továbbfejlesztése, de korlátozásokat is tartalmaz. A mutatók csak nem biztonságos (unsafe) módban használhatóak, és az objektumok nem szabadíthatóak fel közvetlen módon, ezt a szemétgyűjtő (garbage collector) végzi. Sokkal típusbiztosabb nyelv, mint a C++.

A nyelv az objektumorientált programozási paradigmákat követi, amely azt jelenti, hogy osztályokat hozhatunk létre, melyek egységbe zárják az adatokat, és a rajtuk műveleteket végrehajtó metódusokat. Ezekből az osztályokból hozhatunk létre példányokat. Ez azért előnyös számunkra, mert az osztályok által könnyedén tudjuk úgy modellezni a kódunkat, hogy az szorosan tükrözze a valós világ fogalmait. Továbbá az objektumorientált programozás erősen ösztönzi az újra felhasználható kódok írását (code reuse), az öröklődés és a polimorfizmus által. Ez csökkenti a megírandó kód mennyiségét. Ezentúl nagyban megnöveli a kód fenntarthatóságát, mivel a kód nagy mértékben moduláris lesz. Az egységbe zárás által minden osztálynak megvan a saját feladata, így, ha módosítani kell a kódot, azt csak egy helyen kell véghez vinni, így elkerülve a rendezetlen kódból következő hibákat. Emellett kényelmi funkciókkal is rendelkezik, mint például a tulajdonságok (property), melyek lehetővé teszik, hogy az osztályok mezőit az adattagok szintaxisát használva kezeljük.

Hátrányai, közé tartozik, hogy sokszor hiába segíti elő az újra felhasználható kódrészletek írását, emellett sok ismétlődő kódot (boilerplate) is kell írnunk. A .NET keretrendszerből adódóan pedig kevésbé platformfüggetlen, mint más nyelvek, viszont manapság erre is több megoldás elérhető. Emellett a teljesítménye sem a legjobb, mivel a Common Language Runtime (CLR) virtuális gép miatt több memóriát, erőforrásokat használ, mintha natív kódot futtatnánk. [[10](#forras10), [11](#forras11)]

1. *ábra: A C# nyelv logója.* [*[1]*](#forras1)

## 2.2 Visual Studio 2022

A felhasznált fejlesztőkörnyezet a Visual Studio 2022. Ez egy eléggé kézenfekvő választás volt, mivel támogatja a C# nyelvű, WPF appok fejlesztését a .NET keretrendszerben.

Ez az integrált fejlesztőkörnyezet (IDE) számos hasznos funkcióval rendelkezik, mint például fejlett debugging eszközök, melyekkel hatékonyan diagnosztizálhatjuk az esetleges hibákat. Töréspontok elhelyezésével lépésenként futtathatjuk a kódunkat, kisebb részleteket analizálva, így pontosan meghatározhatjuk a hibák forrását. Különböző változók értékeit is figyelhetjük futás közben, mely szintén megkönnyíti a hibák forrásának meghatározását. [[12]](#forras12)

Emellett rendelkezik IntelliSense intelligens kódkiegészítő funkcióval is, mely valós idejű kódelemzést és kódkiegészítést tesz lehetővé, ezzel nagy mértékben megkönnyítve a fejlesztést és gyorsítva a hibamentes kód írását. A kód refactoring funkciók pedig lehetővé teszik a kód könnyű átszervezését, javítják a karbantarthatóságot. A Visual Studio GitHub integrációval is rendelkezik, mely megkönnyíti a verziókövetést és a csapatmunkát. [[13]](#forras13)

Továbbá egy különösen hasznos funkciója a Visual Studio Live Share, mellyel valós időben tud több fejlesztő együttműködni a folyamat során. Lehetővé teszi, hogy megosszuk a fejlesztőkörnyezetünket, mely által egyszerre többen szerkeszthetjük akár ugyan azt a fájlt is, illetve segítsük egymást a hibák keresésében. Ez jelentősen növeli a produktivitást, hiszen azonnal meg tudjuk osztani a kódunkat anélkül, hogy GitHub repository-kat kellene klónoznunk, vagy fájlokat megosztanunk egymással. [[14]](#forras14)

## 2.3 .NET 8.0

A .NET keretrendszer egy, a Microsoft által fejlesztett szoftverfejlesztői platform. Gyors alkalmazásfejlesztést, platformfüggetlenséget és hálózati átlátszóságot támogat. A korábbi, COM nevű platform leváltására fejlesztették ki. Mostanra eszköztára a szoftverfejlesztés számos területét lefedi, mint a kliens-, illetve szerveroldali alkalmazások, adatbáziskezelés, vagy játékfejlesztés.

Alapját a Common Language Infrastructure (CLI) képezi. Ez a szabályok halmaza, melyek leírnak egy nyelvfüggetlen fejlesztői környezetet, futtatókörnyezetet stb. Ennek implementációja a Common Language Runtime (CLR).

A CLI 4 részre bontható:

* Common Language Specification:

A CLS azokat a szabályokat írja le, melyeket a CLI-kompatibilis nyelveknek be kell tartaniuk. A legtöbb .NET nyelv, például a C# is tartalmaz olyan elemeket, amelyek nem felelnek meg ennek, mint például az ulong típusú paraméterek.

* Common Type System:

A CTS a típusokat, azok memóriabeli reprezentációját, és egymással történő interakcióikat írja le.

* Common Language Runtime:

A CLR virtuális gép felel a programok betöltéséért, végrehajtásáért, a kivételkezelésért, memóriamenedzsmentért, illetve a kódbiztonságért is.

* Common Intermediate Language:

A CIL egy köztes kód, minden CLI nyelvű program erre fordul. Ezt futtatáskor egy just in time compiler fordítja natív kódra, melyet a processzor már tud kezelni. Hasonlít a Java bytekódjára. [[15]](#forras15)

## 2.4 WPF (Windows Presentation Foundation)

A WPF egy grafikus felhasználói felületek készítésére alkalmas osztálykönyvtár, melyet szintén a Microsoft fejlesztett. A .NET keretrendszer 3.0-s verziójában jelent meg, és jelentősen különbözik az elődjétől, a Windows Formstól. Választásunk itt azért esett erre, mert itt már sokkal szűkebb volt a lehetőségek száma, hiszen a Windows Forms eléggé elavult, korlátozott.  
Léteznek más alternatívák, mint például az Avalon, vagy a MAUI, viszont ezekben nem rendelkeztünk semennyi tapasztalattal sem, a WPF appok fejlesztésében pedig igen, így ezt választottuk.

A WPF fő újításai közé tartozik a felület és az üzleti logika szétválasztása. A felhasználói felület tervezése az Extensible Application Markup Language (XAML) nyelv segítségével történik, mely egy XML alapú jelölőnyelv. Ez a szétválasztás lehetővé teszi, hogy a fejlesztők, és a felhasználói felület tervezői gördülékenyen együtt tudjanak dolgozni, minimálisra csökkenti az átfedést a feladataik között.

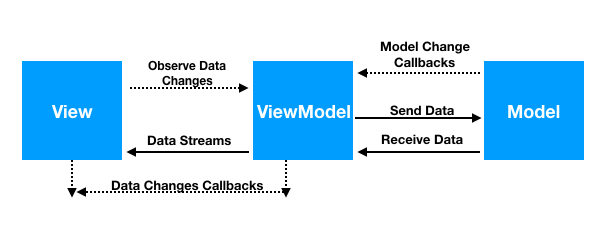
Emellett a WPF grafikus elemei vektorgrafikusak, amely lehetővé teszi, hogy torzítás nélkül tudjunk nagyítani ezeken, továbbá a memóriaigény jelentősen csökken, mivel az nem függ az objektum méretétől, ellenben a raszterképekkel. Ezentúl az alakzatok paraméterei tárolhatóak, és később megváltoztathatóak, így azok módosítása nem megy a pontosság rovására. Ez különösen hasznos a mi esetünkben, hiszen számos grafikai elemmel dolgozunk a programunkban. Ezen grafikus elemek megjelenítéséhez a WPF DirectX-et használ, így növelve a teljesítményt. [[16](#forras16), [17](#forras17), [18](#forras18)]

## 2.5 Model–view–viewmodel

A model-view-viewmodel (MVVM) egy architektúrális minta. Lényege, hogy leválasztja a grafikus felhasználói felületet és az üzleti logikát.

A modell a tartalom állapotát reprezentálja, míg a nézet (view) maga a grafikus felület (GUI), melyet a felhasználó a képernyőn lát. A nézetmodell (viewmodel) a nézet absztrakciója, publikus tulajdonságokat és metódusokat tartalmaz. Az MVC modellel szemben itt a vezérlő helyett a nézetmodell összekötőként szolgál, amely közvetít a nézet és az adatok között. Itt az összekötő funkcióját a XAML tölti be, és segít csökkenteni a megírandó boilerplate kód mennyiségét.

Emellett a tesztelést is elősegíti, mivel az üzleti logika nem függ a felhasználói felülettől, ezáltal sokkal könnyebben tesztelhetővé válik a kód.

 Főként azért esett a választásunk erre az architektúrális mintára, mivel úgy tervezték, hogy jól működjön WPF alkalmazásokkal, segíti a fejlesztést, mind átláthatóság, mind sebesség szempontjából. [[19]](#forras19)

2. *ábra: Az MVVM architektúrája.* [*[2]*](#forras2)

## 2.6 Dependency injection

A Dependency Injection (DI) egy tervezési minta, amely a függőségek kezelését és létrehozását különíti el az osztályoktól, amelyek ezeket használják. Ahelyett, hogy az osztályok maguk hozzák létre a szükséges objektumokat, a DI külső forrásból injektálja őket. Ez a megközelítés segít abban, hogy az osztályok jobban összpontosítsanak a saját feladataikra, nem pedig a függőségek kezelésére.

A DI előnyei közé tartozik a kód újrahasznosíthatóságának és karbantarthatóságának növelése. Mivel az osztályok kevesebb függőséget tartalmaznak, könnyebben módosíthatók és tesztelhetők lesznek. Emellett a DI elősegíti a laza csatoltságot, ami azt jelenti, hogy az egyes komponensek kevésbé függnek egymástól, így a rendszer rugalmasabb és skálázhatóbb lesz. Ez különösen hasznos nagy, összetett projektek esetében, ahol a változtatások gyorsan és hatékonyan végrehajthatók anélkül, hogy az egész rendszert átdolgoznák. [[20]](#forras20)

## 2.7 GitHub

A Git egy nyílt forráskódú elosztott verziókezelő szoftver, amely lehetővé teszi, hogy többen dolgozzanak ugyan azon a projekten, gördülékenyen.

A Git használata által nyomon követhetjük a kódban történt változtatásokat, és több ember külön-külön végzett munkáját tudjuk egyesíteni, oly módon, hogy a kód integritása ne sérüljön. A GitHub ezt a szolgáltatást kínálja a felhőben.

Fő funkciói közé tartozik a verziók nyomon követése, így, ha valami félresikerül, vagy bármilyen oknál fogva szeretnénk visszatérni a szoftver egy korábbi verziójára, ezt könnyedén megtehetjük. Emellett biztosítja az ágak (branch) kezelését, ezáltal elkülönítve tudunk fejleszteni különböző funkciókat, vagy hibákat tudunk kijavítani, majd ezeket az ágakat egyesíteni.

Ezen funkciók által minimalizálni tudjuk a hibák keletkezésének vagy az adatvesztésnek az esélyét. Emellett hatékonyan dolgozhatunk bármekkora méretű projekten, legyen az egyszemélyes vagy nagy fejlesztőcsapatok munkája, minden esetben nagyban növeli a produktivitást. [[21]](#forras21)

# 3. Versenytársak elemzése

A piacon jelenleg számos hasonló webes és asztali alkalmazás található. Ezen alkalmazások előnyeit, illetve hátrányait kielemezve alkotjuk meg a saját alkalmazásunkat. A továbbiakban a jelenleg elérhető legnépszerűbb alkalmazásokat fogjuk elemezni.

## 3.1 Journi

Ebben a webes alkalmazásban egy online tervező segítségével hozhatjuk létre saját fotóalbumunkat. Először hozzá kell adnunk a fotóinkat. Itt maximálisan 1200 darabot adhatunk hozzá, illetve tervezés közben is lehetőségünk van fotókat hozzáadni. A fotók kiválasztása után nyílik meg maga a tervezőfelület. Lehetőségünk van fotókat feltölteni a számitógépünkről, illetve a telefonunkról is. Az alkalmazás felismeri mikor készül a kép, és ezt beszúrja a kép mellé. Többféle album méret közül válaszhatunk, összesen öt darab közül. Különböző stílusokat tudunk beállítani. Laponként van lehetőségünk lapelrendezést is állitani. Lehetőségünk van a lapokra szöveget, dátumot beszúrni, állíthatunk szegélyeket és hátteret is. A szerkesztés befejezése után lehetőségünk van menteni, illetve megrendelni az elkészült fotókönyvet. [[22]](#forras22)

Előnyök:

* Képfeltöltés számitógépről, illetve telefonról is
* Több stílus közül választhatunk
* A dátum automatikusan látszódik a kép mellett

Hátrányok:

* Csak pár előre beállított elrendezés közül választhatunk
* A képen Mobiltelefon, Hordozható kommunikációs eszköz, Mobileszköz, Kommunikációs eszköz látható

  Automatikusan generált leírásA képek egyedi átméretezésére nincs lehetőségünk

3. *ábra: A Journi mobil app felülete.* [*[3]*](#forras3)

## 3.2 CEWE

A CEWE elérhető asztali alkalmazásként, illetve online szerkesztővel is rendelkezik. Ebben az esetben az asztali alkalmazást fogjuk elemezni, mivel a mi szoftverünk is asztali alkalmazás, így nagyobb a hasonlóság a 2 megoldás között, ezáltal relevánsabb az összehasonlítás is. Először ki kell választanunk a papír típusát és a könyvkötést, majd ezután nyílik meg maga a szerkesztőfelület. Itt választanunk kell egy dizájnt.

Lehetőségünk van az oldalakra beszúrni a képeinket akár egyedi elrendezésekben is, a képeket egymásra is tudjuk csúsztatni. Tudunk hátteret állítani akár, oldalanként eltérőt is. Az app opciót nyújt arra is, hogy a fotóinkat szerkesszük. Itt a főbb lehetőségek a körülvágás, tengelyek menti tükrözés, illetve tudunk a fotókra írni is. Lehetőségünk nyílik clip art illusztrációk beszúrására is. Kereteket és maszkokat is állíthatunk be. Az alkalmazás rendelkezik lépés kezelővel, így könnyedén visszavonhatjuk a módosításainkat. Munkánkat egy egyedi fájlkiterjesztésbe (.mcfx) tudjuk menteni a számitógépünkre, hogy később is tudjuk folytatni azt. [[4]](#forras4)

Előnyök:

* Egyedi lapelrendezés
* Fotók szerkesztése
* Esztétikus, átlátható felület
* Jól használható
* Sok funkció

Hátrányok:

* Képek mozgatás nehézkes, ha több van egy oldalon
* Forgatni a képeket csak 90 fokban lehet
* Fótok beszúrása és a háttér módosítása nehézkes

A képen szöveg, képernyőkép, Multimédiás szoftver, szoftver látható

Automatikusan generált leírás4. *ábra: A CEWE online szerkesztő felülete.* [*[4]*](#forras4)

## 3.3 InternetPrint

Az InternetPrint egy internetes online fotókönyv-szerkesztő program. Először ki kell választanunk, hogy kemény vagy puha fedeles legyen a fotókönyvünk. Három lapméret közül tudunk választani. Több fajta oldalelrendezés közül van lehetőségünk választani. A képek keretére tudunk tenni dekorációkat. Előre megszabott hátterekből tudunk válogatni, amiknek a színét is tudjuk állítani. A képek méretezésére, egyedi pozicionálására, szerkesztésére nincs lehetőségünk. A képeket tudjuk feliratozni is. Amikor egy képre szeretnénk szöveget írni, akkor folyamatosan egy felugró ablakba kell begépelnünk a kívánt tartalmat. Ez a megoldás rendkívül kényelmetlen és időigényes, hiszen állandóan ablakok között kell váltanunk, ami jelentősen lassítja a munkamenetet. [[5]](#forras5)

Előnyök:

* Könnyen átlátható felület
* Képeket könnyen és egyszerűen tudunk beszúrni
* Szép dizájn

Hátrányok:

* Nincs lehetőség egyedi elrendezésre
* Csak az előre beállított hátterek közül tudunk választani
* Nincs lehetőség képszerkesztésre, formára vágásra
* Ha írni akarunk egy képre, folyton egy felugró ablakba kell írnunk, ami kényelmetlen
* Kevés szerkesztési funkció
* Kevés, és sablonos lapelrendezési funkciók
* A képen szöveg, képernyőkép, szoftver, számítógép látható

  Automatikusan generált leírásA keretek nem illeszkednek a képekre, belógnak, eltakarják azokat

5. *ábra: Az InternetPrint online szerkesztő felülete.* [*[5]*](#forras5)

## 3.4 MyPrintPix

A rendszer egy sablon kiválasztásával indul, amely meghatározza az online szerkesztő felületének alapstruktúráját. A szerkesztő rugalmas lehetőségeket kínál a lap elrendezésének testreszabására, a képek szabad elhelyezésével. A képek helyét először egy képkeret jelzi, amelynek közepén található a gyakran hibásan működő képfeltöltési gomb.

A vizuális elemek palettájának köszönhetően ikonokat, úgynevezett piktogramokat is adhatunk a lapokhoz, és szabadon mozgathatjuk azokat a kívánt pozícióba.

A képszerkesztő eszközök korlátozott funkcionalitást biztosítanak, mint például a tükrözés, a vágás és az elforgatás. A képekhez kereteket adhatunk, és a háttér is szabadon módosítható. A szerkesztő néhány része magyar, néhány része angol. [[6]](#forras6)

Előnyök:

* Rugalmas, egyedi tervezés
* Könnyen kezelhető
* Átlátható
* Lehetőség van képszerkesztésre
* Testreszabhatóság

Hátrányok:

* Képfeltöltési probléma
* Funkciók néha hibásan működnek
* Nem egységes nyelvű, vannak részek, amik angol nyelvűek

A képen szöveg, képernyőkép, szoftver, képernyő látható

Automatikusan generált leírás6. *ábra: A MyPrintPix online szerkesztő felülete.* [*[6]*](#forras6)

## 3.5 Sooters

Online internetes szerkesztő. Több lapméret közül választhatunk. Lehetőség van a képek szabad elrendezésére és méretezésére. Vannak alap képszerkesztő funkciók, tükrözés, forgatás, méretre vágás. Lehet kereteket hozzáadni a kepékhez. Piktogramok hozzáadására is van lehetőségünk. Egyedi háttér beállítására is van lehetőség, akár oldalankét is egyesével. Van előnézet funkció. Különböző dizájnok beállítására is van opciónk. [[7]](#forras7)

Előnyök:

* Átláthatóság
* Lehetőség egyedi elrendezésre
* Sablonok
* Jól használható
* Sok szerkesztési funkció

Hátrányok:

* Képek minőségét rontja
* Nem lehet saját képet beállítani háttérképnek

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Automatikusan generált leírás7. *ábra: A Sooters online szerkesztő felülete.* [*[7]*](#forras7)

## 3.6 Prodigi

Angol nyelvű online szerkesztő. Egy előre megadott fotókönyvbe tudunk beleilleszteni képeket, csak lapelrendezésekből tudunk váltasztani. Képeket szerkeszteni nem tudunk. Képeket az előre megadott lapelrendezésen kívül máshova helyezni sem tudunk. [[8]](#forras8)

Előnyök:

* Könnyű kezelhetőség
* Átlátható kezelőfelület

Hátrányok:

* Nincs képszerkesztési lehetőség
* Csak előre megadott lapelrendezések vannak
* Nem lehet hátteret állítani

A képen képernyőkép, Multimédiás szoftver, Grafikai szoftver, szoftver látható

Automatikusan generált leírás8. *ábra: A Prodigi online szerkesztő felülete.* [*[8]*](#forras8)

## 3.7 Canva

A Canva egy online képszerkesztő, amelynek több funkciója is van, mi most a témánk miatt a fotókönyv készítő funkcióját elemeztük. Alapvetően a szoftver angol nyelvű, de van hozzá magyar nyelv is, bár maradtak benne angol nyelvű részek. Szép, átlátható, és könnyen kezelhető felülete van. Több stílus közül választhatunk, de egy stílushoz csak egy lapméret tartozik, így ezt nem tudjuk állítani, csak a fizetős verzióval. A képeket egyszerűen drag and drop módon tudjuk hozzáadni, vagy feltölteni egy mappába azokat, amelyeket majd használni akarunk. Vannak képszerkesztő funkcióink, a képeket és a szövegeket oda tudjuk tenni, ahova akarjuk, sőt, ha egy elemet nem akarunk véletlen elmozdítani, akkor azt tudjuk zárolni, hogy ne lehessen elmozdítani. Rengeteg grafikát és díszitőelemet tudunk hozzáadni, tudunk rajzolni is a lapokra. Oldalszámozás is elérhető az alkalmazásban. Egyes képekhez tudjuk igazítani az oldal színét. Mivel bárhova tudjuk tenni a képeinket, ezért előfordul, hogy egymásra is csúznak a képeink, de a rétegek kezelésére is van beépített funkció. Szerkesztés közben a felület animálva is van, ami fokozza a felhasználói élményt. Lehetséges videót vagy hangot is hozzáadni a fotókönyvhöz, ezek értelemszerűen csak digitális formában érhetőek el.

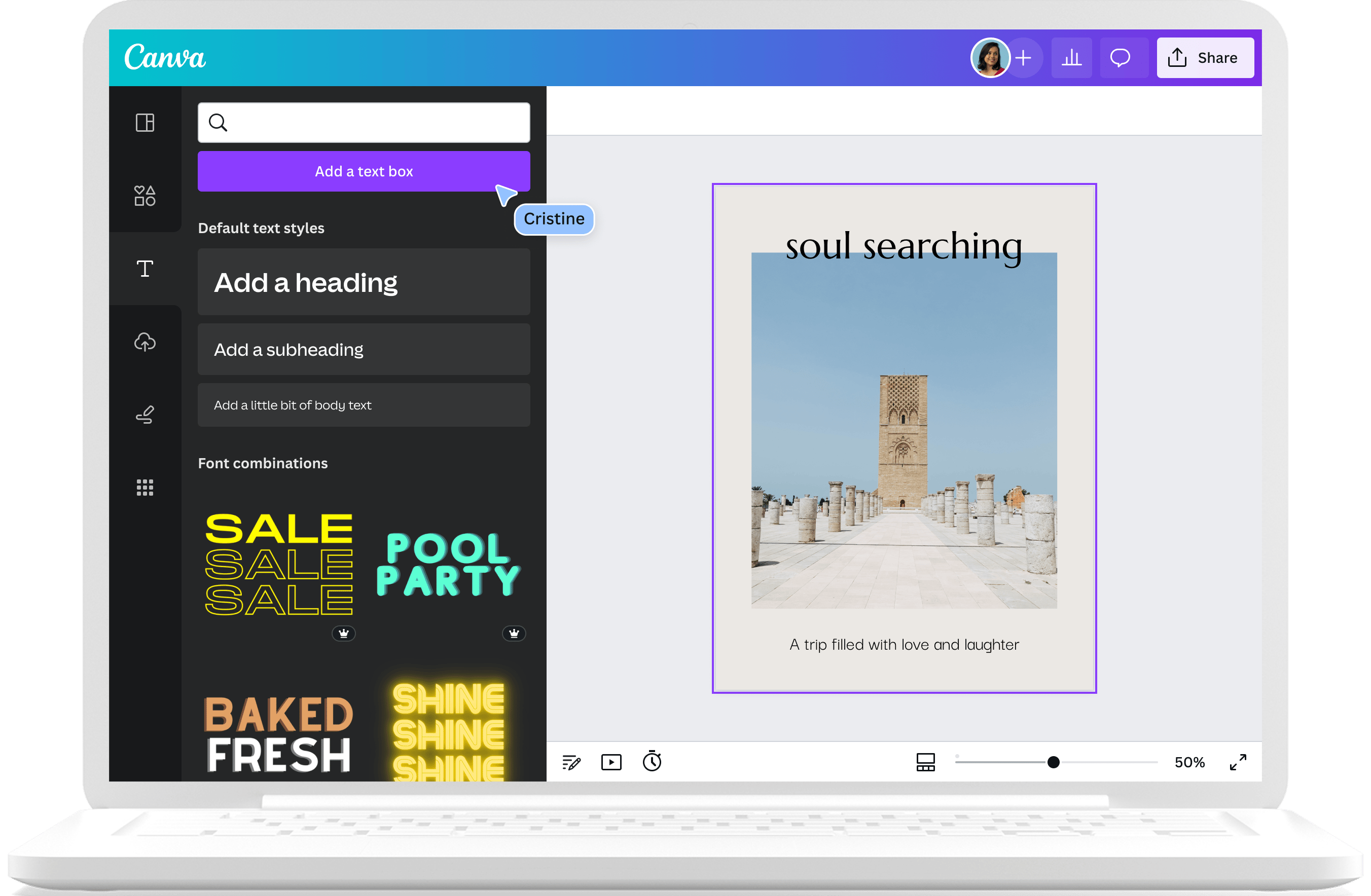
Munkáinkat több formátumban is tudjuk menteni. Lehet hozzáadni táblázatokat, 3D-s grafikákat is. Tartalmaz egy varázslót is, ami segít a szerkesztésben és még AI funkciókat is tartalmaz. A termék alapvetően ingyenes, de van fizetős verziója, amiben további témákat és díszitőelemeket kapunk, illetve a videókról eltávolítja a vízjeleket, de mi csak az ingyenes verziót teszteltük. [[23]](#forras23)

Előnyök:

* Felhasználó barát kezelő felület.
* átlátható.
* Mentésre több formátumba is lehetséges.
* Videók hangok hozzá adása ezzel egy digitális fotókönyvet létre hozva .
* Varázsló és AI funkciók.
* Rengeteg diszitő elem és téma.

Hátrányok:

* A magyar fordításban maradtak angol szövegek.
* Az ingyenes verzióban nem minden elérhető
* Lap méretet nem lehet egyedién állítani csak a fizetős verzióval.

9. *ábra: A Canva online szerkesztő felülete.* [*[9]*](#forras9)

## 3.8 Táblázatos összehasonlítás

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Journi** | **CEWE** | **InternetPrint** | **MyPrintPix** | **Sooters** | **Prodigi** | **Canva** |
| **Platform** | Online | Asztali, Online | Online | Online | Online | Online | Asztali, Online |
| **Testreszabhatóság** | Közepes | Magas | Alacsony | Magas | Magas | Alacsony | Nagyon magas |
| **Elrendezési lehetőségek** | Korlátozott | Magas | Korlátozott | Magas | Magas | Korlátozott | Nagyon magas |
| **Képszerkesztés** | Alapvető | Közepes | Nincs | Alapvető | Alapvető | Nincs | Közepes |
| **Háttér opciók** | Korlátozott | Magas | Korlátozott | Magas | Magas | Korlátozott | Nagyon magas |
| **Szöveg testreszabás** | Alapvető | Közepes | Alapvető | Közepes | Közepes | Alapvető | Magas |
| **Használhatóság** | Magas | Közepes | Magas | Közepes | Magas | Magas | Magas |
| **Egyedi jellemzők** | Automatikus dátum beszúrás | Lépésenkénti folyamat | Egyszerű felület | Piktogramok | Sablonok | Csak angol | MI funkciók |

## 3.9 Összefoglalás

A legtöbb fotókönyv szerkesztő online elérhető, ugyanakkor találtunk asztali alkalmazási is, és egy adott cég termékeként funkcionál. Ezek a felületek lehetővé teszik a fotókönyv teljes testreszabását a rendeléséig. Bár a funkciókban némi eltérés tapasztalható, általában lehetőség van az oldalelrendezés módosítására, akár egyedi elrendezések létrehozására is. Alap képszerkesztő funkciók, mint az elforgatás, tükrözés vagy a kivágás is általában elérhetőek, és sok esetben a háttér is személyre szabható. Bár néhány alkalmazásnál tapasztalható funkcióbeli hiányosság, vagy funkciók nem megfelelő működése, de többségében ezek a szoftverek átláthatóak és könnyen használhatóak.

# 4. Funkciók ismertetése

A következő fejezetben az alkalmazás funkciói kerülnek részletes bemutatásra, a felhasználók szemszögéből.

## 4.1 Felhasználói felület

Az alkalmazás használata nincs felhasználói fiókokhoz kötve, így indítás után nem kell bejelentkeznünk, hanem egyből a felhasználói felület fogad minket. Ez a felület 4 fő komponensből áll.

### 4.1.1 Navigációs sáv

A navigációs sávban a más asztali alkalmazásokból is megszokott fájl, illetve szerkesztés menüket találhatjuk meg. Az itt található funkciók egy része akkor is elérhető, ha nincs jelenleg megnyitott projekt.

10. *ábra: A navigációs sáv.*

### 4.1.1.1 A fájl menü

A fájl menü tartalmazza a projektek és az alkalmazás manipulálásával kapcsolatos alapvető funkciókat. Itt található az új, üres projekt megnyitása. Erre az opcióra kattintva egy új, üres dokumentum nyílik meg, 1 darab üres oldallal. Az oldal tájolása álló, mérete megegyezik az A4-es lap méretével.

A megnyitás menüpontra kattintva a fájlkezelő ablaka nyílik meg, ahol kiválaszthatjuk a már korábban elmentett .proj kiterjesztésű projektfájlunkat, melyet szeretnénk betölteni. A projekt kiválasztása után az betöltődik minden tulajdonságával, oldalakkal, képekkel stb. Ezt követően az alkalmazás címe a megnyitott projekt elérési útja lesz.

A mentés opciót választva szintén a fájlkezelő ablaka nyílik meg, ahol megadhatjuk a mentés helyét, és a fájlnevet. Ha a projekt, amelyen jelenleg dolgozunk, már mentésre került, akkor a mentés opció ezt a fájlt fogja felülírni. Mentést követően is megváltozik az alkalmazás címe, a projekt elérési útjára.

Amennyiben nem szeretnénk felülírni a már elmentett projektünket, akkor a mentés másként opciót választva szintén felugrik a fájlkezelő ablaka, ahol kiválaszthatjuk a mentés helyét és a fájlnevet.

Ezen opció kiválasztása mindig új fájl létrehozását eredményezi, így elkerülve, hogy felülírjuk a már elmentett projektfájlunkat. Ezt az opciót választva is megváltozik az alkalmazás címe.

Ez alatt a menüpont található a bezárás opciója. Erre kattintva az alkalmazás bezárja a jelenleg nyitott projektet, az alkalmazás címe pedig vissza változik az eredeti szövegre. Ha vannak nem mentett módosításaink, ez az opció ezt nem jelzi a felhasználó felé, így mindenképpen ügyeljünk a mentésre bezárás előtt.

A következő menüpont az exportálás menüpontja. Erre kattintva szintén a fájlkezelő ablaka ugrik fel, ahol megadhatjuk az elérési utat és a fájlnevet. Ez az opció PDF formátumban menti a projektet. Ezután találjuk a nyomtatás opcióját. Erre kattintva felugrik a nyomtatási ablak, ahol kiválaszthatjuk a használni kívánt nyomtatót, és kinyomtathatjuk a projektünket.

A képen szöveg, képernyőkép, képernyő, szoftver látható

Automatikusan generált leírás11. *ábra: A nyomtatási ablak.*

A fájl menü utolsó menüpontja a kilépés. Erre kattintva az alkalmazás bezáródik. Itt is fontos ügyelnünk arra, hogy mentsük a haladásunkat, mivel ez az opció sem figyelmeztet, hogy el nem mentett változtatásaink vannak.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás12. *ábra: A fájl menü.*

### 4.1.1.2 A szerkesztés menü

A navigációs sávon a fájl menü mellett a szerkesztés menüt találhatjuk meg. Itt találhatóak meg az oldalakkal kapcsolatos műveletek.

Az új oldal opcióra kattintva a jelenleg megnyitott dokumentumhoz egy új oldalt adunk hozzá. Az oldal alapértelmezett tájolása álló, mérete az A4-es oldal méretével egyezik meg.

A kijelölt oldal törlése opció a jelenleg kijelölt oldalt törli a dokumentumból. Erre az opcióra kattintva felugrik egy figyelmeztető ablak, mivel ez a művelet nem visszavonható. Itt megerősíthetjük az oldal törlését. Ez az opció csak akkor használható, ha legalább 2 oldallal rendelkezik a jelenleg megnyitott projektünk.

A képen szöveg, képernyőkép, sor, Betűtípus látható

Automatikusan generált leírás13. *ábra: A törlést megerősítő felugró ablak.*

A szerkesztés menüpont következő opciója a kijelölt oldal elforgatása. Erre kattintva a kijelölt oldal tájolása az ellenkező tájolásra változik.

Végül itt találhatjuk a kijelölt oldal mentése sablonként opciót. Ezt kiválasztva a fájlkezelő ablaka felugrik, és itt megadhatjuk a fájlnevet és az elérési utat. A fájl kiterjesztése .template. A sablonban mentésre kerül az oldal háttérszíne, a szövegmezők és azok tulajdonságai, a szövegbuborékok, illetve a képek pozíciója és tulajdonságai.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás14. *ábra: A szerkesztés menü.*

A fájl és szerkesztés menüpontok a navigációs sávon megtalálható egy mentés ikon is, melynek funkcionalitása megegyezik a fájl menüben található mentés opcióéval. Megtalálhatjuk itt továbbá a visszavonás, és a visszavonás megismétlése funkciókat is.

### 4.1.2 Bal oldali panel

A jelenleg megnyitott projekt bal oldalán található egy panel, mely mindig aktív, és az alkalmazás fő funkcióit tartalmazza. Itt található a varázsló, a képek hozzáadása, képkeretek kiválasztása, szövegdobozok hozzáadása, szövegbuborékok hozzáadása, és az oldalelrendezések menüje is. Az itt található funkciók közül semelyik sem működik jelenleg megnyitott projekt hiányában.

A képen Színesség, Grafika, Grafikus tervezés, rajz látható

Automatikusan generált leírás15. *ábra: A bal oldali panel gombjai.*

### 4.1.2.1 Varázsló

A varázsló funkció gombját egy varázspálca ikon jelzi. Erre kattintva érheti el a felhasználó a varázslót, mely a fotókönyvek automatikus elkészítésében nyújt segítséget.

A varázsló funkció képes beállítani az oldalak és a szövegmezők háttérszínét, és képeket helyez el a helykitöltő képek helyére. A gombra kattintva először a fájlkezelő ablak nyílik meg. Itt tetszőleges számú képet választhatunk ki. Amennyiben találhatóak helykitöltő képek bármely oldalon, a varázsló ezeket kicseréli a kiválasztott képekre. Minden kiválasztott képet pontosan egyszer helyez el a projektben, ha pedig több képet választunk ki, mint ahány helykitöltő található, akkor a kijelölt képekből az első megfelelő számút választja ki.

Minden képhez a képarányban legjobban illeszkedő helykitöltőt választja ki, és a „fekete sávok” effektussal cseréli ki őket. A sávok tényleges színét a kép színéből határozza meg.

Ezután a varázsló minden oldalhoz kiválasztja a képek alapján a háttérszínt, illetve ennek a komplementer színét az oldalon elhelyezkedő szövegmezőkre alkalmazza. Ha nincs helykitöltő az projektünkben, akkor ezt ettől függetlenül is elvégzi a varázsló, viszont a fájlkezelő ablaka nem nyílik meg.

A képen Téglalap, sor, Párhuzamos, diagram látható

Automatikusan generált leírás16. *ábra: A helykitöltő képek kinézete.*

### 4.1.2.2 Képek hozzáadása

A képek hozzáadásának funkciója megnyitja a fájlkezelő alkalmazást, és a kiválasztott képet hozzáadja a kijelölt oldal bal felső sarkához. Amennyiben nem jelöltünk ki egy oldalt, akkor a legutolsó oldalhoz adja hozzá a kiválasztott képet.

### 4.1.2.3 Képkeretek hozzáadása

A képkeretek hozzáadásának gombjára kattintva egy felugró ablak jelenik meg, ahol a választható képkeretek közül válogathatunk. Ezt az ablakot az alján található mégse gombbal zárhatjuk be.

Ha van jelenleg kijelölt kép, és a felugró ablakban is kijelöltük az alkalmazni kívánt képkeretet, akkor a mentés gombra kattintva a képkeretet alkalmazhatjuk a kijelölt képre. Ha ezen feltételek közül bármelyik nem teljesül, akkor a mentés gomb nem csinál semmit. Ha a kijelölt kép már rendelkezik képkerettel, akkor az új képkeret felülírja a már meglévőt.

A képen Téglalap, sor, képernyőkép, tér látható

Automatikusan generált leírás17. *ábra: A képkeret választó felugró ablak.*

### 4.1.2.4 Szövegdobozok hozzáadása

A képen szöveg, képernyőkép, Betűtípus, sor látható

Automatikusan generált leírásA szövegdobozok hozzáadásának gombjára kattintva megjelenik egy felugró ablak, ahova beírhatjuk a hozzáadni kívánt szövegdoboz tartalmát. A felugró ablak alján található pipa gombbal hozzáadhatjuk a szövegdobozt az aktív oldalhoz, vagy ha ilyen nincs, akkor a legutolsó oldalhoz. Ha nem adunk meg szöveget a felugró ablakban, akkor a szövegdoboz nem kerül hozzáadásra. A szöveg alapértelmezett betűmérete 12-es, betűtípusa Segoe UI, stílusa nincs, színe fekete.

18. *ábra: A szövegdoboz hozzáadásának felugró ablaka.*

### 4.1.2.5 Szövegbuborékok hozzáadása

A szövegbuborékok hozzáadásának felugró ablaka megegyezik a képkeretek felugró ablakával. Ezt a funkciót választva megjelenik a felugró ablak, ahol kiválaszthatjuk a hozzáadni kívánt szövegbuborékot, melyet vagy a kijelölt, vagy az utolsó oldalhoz adja hozzá az alkalmazás.

A képen diagram, vázlat, Betűtípus, clipart látható

Automatikusan generált leírás19. *ábra: A szövegbuborék választó felugró ablak.*

### 4.1.2.6 Sablon oldalak hozzáadása

A projektünkhöz sablon oldalakat adhatunk hozzá, melyek helykitöltő képeket tartalmaznak. Itt is egy hasonló felugró ablak fogad minket, mint a képkeretek és szövegbuborékok esetében. Az előre meghatározott sablonok előnézeti képét láthatjuk itt. Ebben az ablakban választhatunk az előre meghatározott elrendezések között, illetve beimportálhatunk saját sablonokat is. Utóbbi esetben felugrik a fájlkezelő ablaka, és kiválaszthatjuk a sablon elérési útját. A sablon a projekt legvégére kerül hozzáadásra. Saját sablon esetén a sablonban megtalálható képek a helykitöltő képre cserélődnek.

A képen diagram, sor, Téglalap, szimbólum látható

Automatikusan generált leírás20. *ábra: A sablon választó felugró ablak.*

### 4.1.2.7 Színválasztó

A bal oldali panelen találhatjuk meg az elemek színének manipulálására szolgáló gombokat. Itt néhány előre meghatározott szín közül választhatunk, illetve elérhető egy gomb is, mellyel részletesen megadhatjuk a kívánt színt. Csúszkával választhatjuk ki a szín tónusát, és egy skálán adhatjuk meg az árnyalatot. A folyamat során folyamatos vizuális visszajelzést kapunk a szín kinézetéről. Megadhatjuk továbbá a színt hexadecimális vagy RGB kód megadásával is. A szín a kijelölt szövegmező tartalmának színére, vagy a kijelölt oldal hátterére kerül alkalmazásra.

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Automatikusan generált leírás21. *ábra: A színválasztó.*

### 4.1.3 Fő szerkesztő felület

Az alkalmazás közepén helyezkedik el a fő szerkesztő felület, ahol a jelenleg megnyitott dokumentumunkat tudjuk szerkeszteni. Itt láthatóak az oldalak, azokon elhelyezkedő képek, szövegdobozok stb.

A dokumentum tetején megtalálható egy menüsáv, melyen a dokumentum nézetét változtathatjuk. Elérhetjük a nyomtatási funkciót is innen, illetve nagyíthatunk, kicsinyíthetünk a dokumentumon, és választhatunk előre meghatározott oldal megjelenítések között is.

Elérhető 100%-os nagyítás, teljes oldalszélesség, teljes oldalmagasság, és 2 oldal megjelenítése is. A menüsávon elhelyezkedő opciók rendelkeznek billentyűkombinációkkal is.

A nyomtatást a Ctrl + P, a nagyítást A Ctrl +, a kicsinyítést a Ctrl -, a 100%-os nagyítást Ctrl + 1, a teljes oldalszélességet Ctrl + 2, a teljes oldalmagasságot Ctrl + 3, és a 2 oldal megjelenítését a Ctrl + 4 billentyűkombináció segítségével érhetjük el.

A képen képernyőkép látható

Automatikusan generált leírás22. *ábra: A dokumentum megjelenítő.*

A szerkesztő felületen a jelenleg kijelölt oldal a tetején, illetve az alján villogó piros sávok jelzik. Egy oldalt úgy jelölhetünk ki, hogy belekattintunk egy üres részén, ahol semmilyen más elem nem helyezkedik el.

Az oldalon elhelyezkedő elemeket egérkattintással jelölhetjük ki. Ekkor megjelenik körülöttük egy szaggatott vonal, melynek a 4 sarkán 4 fogópont helyezkedik el, melyek segítségével egérkattintással majd húzással méretezhetjük az elemeket.

A szövegmezőkre történő dupla kattintás után felugrik ugyan az az ablak, mely a szövegmezők hozzáadásakor jelenik meg, és itt szerkeszthetjük a szöveges tartalmat.

Minden elem rendelkezik egy kontextusfüggő menüvel, melyet jobb egérkattintással érhetünk el. Itt található az elem törlésének opciója. Képelemek esetében 2 további opcióval bővül ez a menü: kép módosítása, keret törlése.

A képen képernyőkép látható

Automatikusan generált leírás23. *ábra: Egy kijelölt elem, és a kontextusmenüje.*

A kép módosítása opcióra kattintva felugrik a fájlkezelő, ahol kiválaszthatjuk az új képet, melyet a régi helyére szeretnénk beszúrni. Miután kiválasztottuk az új képet, felugrik egy ablak, ahol a módosítás típusát adhatjuk meg. 3 opció közül választhatunk: eredeti méret megtartása, kitöltés nagyítással, fekete sávok.

Az eredeti méret megtartása opció esetén az alkalmazás automatikusan átméretezi az új képet a régi képpel megegyező méretűre.

A kitöltés nagyítással opció akkorára nagyítja fel a képet, hogy az kitöltse a régi kép helyét, a kilógó részeket pedig levágja.

A fekete sávok opció is átméretezi a képet, akkorára, hogy az beférjen a régi kép helyére kilógás nélkül, az új kép méretarányát megtartva. Az üresen maradt helyet pedig a kép színe alapján számított színnel tölti ki.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás24. *ábra: Kép módosításának felugró ablaka.*

A keret törlése opció eltávolítja az adott képről a képkeretet, ha az rendelkezik eggyel.

### 4.1.4 Jobb oldali kontextusfüggő panel

A képernyő jobb oldalán helyezkedik a panel, mely az elemek manipulálásának főbb opcióit tartalmazza. Ez a panel csak abban az esetben jelenik meg, ha rákattintunk egy elemre a jelenleg aktív projektben, és eltűnik, ha kikattintunk belőle.

A panel legtetején helyezkedik el a forgatás opciója. Itt egy szövegmezőben adhatjuk meg a forgatás mértékét, fokban mérve, 0 és 360 között.

A forgatás alatt helyezkedik el 2 gomb, mellyel a kijelölt elemet tükrözhetjük, az X vagy Y tengelyek mentén. Egyszerre akár mindkét tükrözést és forgatást is alkalmazhatunk egy elemre.

A tükrözés alatt helyezkedik el az elem pozícionálása, melyet az X és Y koordináták megadásával módosíthatunk. Ezeket a koordinátákat szövegmezőkben adhatjuk meg, és a maximális értékeik az oldal tájolásától függően az A4-es oldal méreteihez vannak kötve, minimális értékük pedig 0. Ezen beviteli mezők által pontosabban adhatjuk meg az elem pozícióját, mintha azt az egérrel tennénk.

A pozícionálás alatt ugyancsak 2 szöveges beviteli mezőt találhatunk, melyekkel az elemek szélességét és magasságát adhatjuk meg, pixelekben mérve, ezáltal szintén pontosabban módosíthatjuk az elem méreteit, mintha egér húzásával tennénk azt. Ezeknek a beviteli mezőknek nincs maximális értéke, minimális értékük pedig 0. A 2 beviteli mező alatt találhatunk egy jelölőnégyzetet, mellyel azt adhatjuk meg, hogy az átméretezés során kívánjuk-e megtartani az elem eredeti méretarányát. Ez a beállítás csak a beviteli mezőkkel történő átméretezésre vonatkozik, az egérrel történő módosításra nem.

Képelemek esetében végül a rétegek kezelésével kapcsolatos 2 gombot találhatjuk itt. Ugyan új rétegeket nem hozhatunk létre, a meglévő rétegen belül az elemek egymás feletti elhelyezkedését az előre-, illetve hátraküldés gombok segítségével változtathatjuk meg. Az előreküldés gomb legelőre, a hátraküldés pedig leghátulra küldi az adott elemet, ezzel biztosítva az elemek fedésével kapcsolatos módosításokat.

Ha az éppen kijelölt elemünk egy szövegmező, akkor ez a panel több opcióval is bővül. Ezek közül az első a betűméret módosítása, melyet szintén szöveges beviteli mezőben módosíthatunk. 1-esnél kisebb értéket nem adhatunk meg, maximális érték nincs.

Ezalatt egy legördülő listában választhatjuk ki a szöveg betűtípusát. Alapértelmezetten ez Segoe UI, és az elérhető opciók listáját a rendszerben elérhető betűtípusok képezik.

A képen szöveg, elektronika, képernyőkép, szoftver látható

Automatikusan generált leírás25. *ábra: Betűtípus választó legördülő lista.*

A képen szöveg, képernyőkép, szám, képernyő látható

Automatikusan generált leírás Ezalatt a lenyíló lista alatt 3 gomb helyezkedik el, melyekkel a szöveg stílusát módosíthatjuk. Az első gombbal a szöveget félkövérré tehetjük, a második gomb a dőlt betűstílus beállításáért felel, a harmadikkal pedig aláhúzottá tehetjük a szöveget.

26. *ábra: A panel, egy szöveges mező esetében.*

# 5. Megvalósítás

A következő fejezet során az eddig leírt funkciók programozási hátterét fogjuk bemutatni részletesen, kódrészletek segítségével.

## 5.1 App.xaml.cs

Az alkalmazás indítását követően az App.xaml.cs fájlban példányosítjuk a megfelelő nézetmodelleket, és regisztráljuk a függőségi injektálási konténerben a megfelelő tárolókat, nézeteket és nézetmodelleket, ezáltal biztosítva, hogy a nézetmodelljeink hozzáférjenek ezekhez, és a függőségi injektálás automatikusan kezeli ezeknek az életciklusát.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás27. *ábra: Az App.xaml.cs konstruktora.*

## 5.2 MainWindow.xaml

Mivel az MVVM mintát alkalmazzuk, de valójában az alkalmazás nem valósít meg semmiféle navigációt a felugró ablakokon kívül, így indításkor megjelenítjük a fő nézetet, melyet később nem is módosítunk. A fő nézetünk a MainWindow.xaml, melyen belül állítjuk be a LayoutView nézetet. Ez a LayoutView tartalmazza a navigációs sávot, és ezen belül is beállítjuk az EditorLayoutView nézetet, mely tartalmaz 2 komponenst.

Ezek a Toolbar, mely a bal oldali panel elrendezését definiálja, és az Editor, mely magában tartalmaz egy DocumentViewer elemet, mely a projekt megjelenítéséért felelős, és a PropertiesPanel komponenst, mely a jobb oldali panel elrendezését definiálja.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás28. *ábra: A MainWindow.xaml részlete.*

## 5.3 Tárolók

### 5.3.1 NavigationStore

A NavigationStore felelős az aktuálisan használt nézetmodell tárolásáért és a változásairól történő értesítésért. Az CurrentViewModel tulajdonság tárolja az aktuális nézetmodellt, míg az OnCurrentViewModelChanged metódus értesíti a felhasználói felületet, hogy új nézetet kell megjeleníteni. Minden egyes nézetmodell váltásakor az előző nézetmodell Dispose() metódusát is meghívja, ezáltal biztosítva a memóriakezelést. Jelen esetben ez csak az iniciális navigációhoz szükséges.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás29. *ábra: A NavigationStore osztály.*

### 5.3.2 ModalNavigationStore

A ModalNavigationStore osztály ugyan azt a funkciót valósítja meg, mint a NavigationStore osztály, csak a felugró ablakok esetében.

### 5.3.3 SelectedElementStore

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírásA SelectedElementStore osztály a jelenleg kiválasztott elem nyomonkövetéséért felelős. Különböző eseményeket tartalmaz, melyekre a megfelelő nézetmodellekben feliratkozunk, ezáltal értesítve az osztályt, hogy a kiválasztott elem megváltozott.

30. *ábra: A SelectedElementStore osztály egy részlete.*

### 5.3.4 ProjectStore

A ProjectStore osztály tartalmazza a jelenleg megnyitott projektet, és ezáltal az azzal kapcsolatos műveleteket is. Itt tároljuk a visszavonás és a visszavonás megismétléséhez szükséges vermeket, a jelenlegi projekt elérési útját, illetve tartalmaz egy SelectedElementStore osztálypéldányt is. A konstruktorban beállításra kerül a SelectedElementStore példánya, és az *\_undoStack* és *\_redoStack* vermek is példányosításra kerülnek.

A NewProject metódussal nyithatunk új, üres projektet, a *CurrentProject* változó értékének egy új Project osztálypéldányra változtatásával. A CloseProject metódus null értékre állítja a *CurrentProject* változó értékét. Mivel ezek a metódusok a *CurrentProject* változó értékét módosítják, és a változó setter-je meghívja az OnCurrentProjectChanged metódust, mely a CurrentProjectChanged eseményt idézi elő, így a többi nézetmodell is értesül a projekt változásáról.

A NewPage metódus a Project osztály AddBlankPage metódusát hívja meg, majd egy NewPageAdded eseményt idéz elő, ezáltal értesítve a szükséges nézetmodelleket.

A SerializeDocument metódus egy FixedDocument típusú paramétert vár, és a dokumentum XML formátumba történő mentéséért felel. A metódus első lépésben eltávolítja a kontextus menüt a dokumentumból a RemoveContextMenu metódus segítségével. Ez azért szükséges, mivel az XamlWriter nem képes szerializálni ezeket. Ezt követően létrehoz egy új XDocument objektumot, amely a dokumentum XML reprezentációját tartalmazza.

A dokumentum tartalmának mentése érdekében az XamlWriter.Save metódus használatával a FixedDocument objektumot egy XML író segítségével szerializálja. A folyamat során minden egyes oldal (ha az FixedPage típusú) és azok gyermekelemei is ellenőrzésre kerülnek. A gyermekelemek között, ha találkozik egy AdornerDecorator típusú elemmel, azt a megfelelő FrameworkElement-té alakítja.

Ezután az AdornerLayer réteg segítségével a metódus az aktuális elemhez tartozó adorning információkat gyűjti össze. Ha bármilyen FrameAdorner típusú adorner található, annak állapotát, például a hozzá tartozó elem címkéjét és a forrást, egy FrameAdornerState objektumban tárolja. Ez a FrameAdornerState egy helper osztály, melynek segítségével a képkeretekkel kapcsolatos informáicókat tároljuk. Az így létrehozott adatokat az XML dokumentum megfelelő részébe építi be, hogy azok később visszaállíthatók legyenek.

Végül a metódus visszaállítja a kontextus menüt a RecreateContextMenu metódus hívásával, és a dokumentum XML reprezentációját visszaadja szöveges formátumban. Ha bármilyen hiba történik a folyamat során, a metódus null értéket ad vissza.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás31. *ábra: A SerializeDocument metódus egy részlete.*

A DeserializeDocument metódus egy XML formátumban tárolt dokumentum állapotot vár, és azt deszerializálja vissza egy FixedDocument típusú objektummá. A metódus első lépésében a bemeneti documentState sztringet XML dokumentummá alakítja az XDocument.Parse metódus segítségével, majd az XML gyökér eleméből kinyeri a FixedDocument reprezentációját.

Az XamlReader.Load metódus segítségével az XML-ből egy FixedDocument objektumot hoz létre. Ezután a metódus törli a korábbi adorning információkat a ClearAdorners metódus meghívásával.

A deszerializálás során minden egyes FrameAdornerState elem feldolgozásra kerül az XML-ben. Minden FrameAdornerState elem tartalmazza az "AdornedElement" és "SourceUri" értékeket, amelyek a képkerethez tartozó elemet, és a keret forrását adják meg. Ezeket az adatokat egy FrameAdornerState objektumba helyezi, amely később a megfelelő oldalon és képen található adornerekhez van rendelve.

Minden oldal és gyermekeleme át van vizsgálva, hogy az adott Image típusú elem megfelel-e az adorner állapotának. Ha találkozunk egy megfelelő képpel, akkor az adorner információk (például a kép és a hozzá tartozó forrás URI) hozzáadásra kerülnek a deszerializált adorner listához.

A végén a metódus elkészíti az elemek kontextus menüit a RecreateContextMenu metódus segítségével, majd visszaadja a teljes deszerializált FixedDocument objektumot. Ha bármilyen hiba történik a deszerializálás során, a metódus null értéket ad vissza.

A képen szöveg, képernyőkép, Betűtípus, dokumentum látható

Automatikusan generált leírás32. *ábra: A DeserializeDocument metódus egy részlete.*

A RecreateContextMenu metódus a CreateTextBlockContextMenu és CreateImageContextMenu metódusok segítségével elkészíti az elemek kontextus menüit. Ezek a metódusok az adott elem típusa alapján hozzáadják a megfelelő opciókat a menühöz.

Az AddAdorners metódus az aktuális dokumentumban található oldalakra adja hozzá a szükséges adorning elemeket. A metódus először ellenőrzi, hogy van-e bármilyen adorner a listában, és ha nincs, akkor visszatér anélkül, hogy bármit tenne.

Ha az adorner lista nem üres, a metódus végigiterál a listában található minden egyes adorneren. Minden adorner három elemből áll: az első elem a célzott oldal azonosítóját tartalmazza, a második egy Image típusú elem, amelyhez az adorner tartozik, a harmadik pedig a BitmapImage, amely az adornerhez tartozó képet képviseli.

A metódus ezután végigmegy a projekt dokumentumának minden oldalán. Ha egy oldal egy FixedPage típusú objektumot tartalmaz, akkor összeveti az oldal címkéjét (Tag) az adorner első elemével. Ha az oldal címkéje megegyezik az adorner azonosítójával, akkor ellenőrzi, hogy az adorner már hozzá van-e adva a megfelelő AdornerLayer réteghez.

Ha az adorner még nem található meg a FixedPage rétegében, és nem találkozik FrameAdorner típusú adornerrel, akkor létrehoz egy új FrameAdorner példányt és hozzáadja azt az AdornerLayer réteghez.

A képen szöveg, képernyőkép, Betűtípus, sor látható

Automatikusan generált leírás A metódus célja, hogy minden adorner megfelelő elemhez legyen rendelve a dokumentumban, és szükség esetén új adorner példányokat hozzon létre, ezáltal biztosítva, hogy a képkeretek a megfelelő elemekhez kerülnek hozzáadásra.

33. *ábra: Az AddAdorners metódus.*

A SaveProject metódus egy projekt állapotát menti el a megadott fájl elérési úton. A metódus első lépésében ellenőrzi, hogy a CurrentProject objektum és annak Document tulajdonsága érvényesek-e. Ha bármelyik értéke null, akkor a metódus nem végez el semmilyen műveletet, és visszatér.

Ha a projekt és a dokumentum érvényesek, akkor a metódus a SerializeDocument metódust hívja, hogy a dokumentum aktuális állapotát XML formátumban tárolja. Az így kapott dokumentumállapotot a System.IO.File.WriteAllText segítségével menti el a megadott filePath helyre.

A mentés után a CurrentProjectFilePath változó értékét frissíti, hogy az aktuális fájl elérési útja mindig elérhető legyen a további műveletekhez.

A metódus egy try-catch blokkot tartalmaz, hogy a hiba esetén a program ne omoljon össze, azonban a catch blokkban jelenleg nem végzünk semmiféle hibakezelési műveletet.

A LoadProject metódus egy projektet tölt be a megadott fájlból, és visszaállítja a projekt dokumentumának állapotát. A metódus először ellenőrzi, hogy a fájl létezik-e a megadott elérési úton (filePath). Ha a fájl nem található, a metódus visszatér anélkül, hogy bármit is tenne.

Amennyiben a fájl létezik, a metódus beolvassa a fájl tartalmát, és azt egy sztringként tárolja. A sztringet ezután deszerializálja a DeserializeDocument metódus segítségével. Ha a deszerializálás során hiba történik, vagy ha a deszerializált dokumentum értéke null, akkor a metódus nem folytatódik tovább.

Ezután a deszerializált dokumentum oldalainak ellenőrzése következik. A metódus először meghatározza azokat az oldalakat, amelyek szélessége nagyobb, mint a magasságuk (ezeket az oldalakat elforgatták). Ezen oldalak indexeit egy listába gyűjti, hogy azokat később megfelelően kezelni lehessen.

Ezután, ha a CurrentProject még nem létezik, létrehozza azt. Ha az első oldal el van forgatva, akkor az új projektet forgatásra készen (azaz true értékkel) hozza létre. Ezt követően az összes további oldalt is hozzáadja az új projekthez, figyelembe véve, hogy azok el vannak-e forgatva, és ha igen, akkor az új oldalakat az előzőekhez hasonló módon hozza létre.

A deszerializálás után a metódus minden oldal gyermekelemeit és háttérszínét átállítja a deszerializált dokumentumban tárolt értékekre. Az oldalakat és azok gyermekelemeit átmásolja a deszerializált dokumentumból a jelenlegi projektbe, biztosítva, hogy a tartalom és a struktúra is pontosan úgy jelenjen meg, mint a mentés előtt.

Végül a metódus frissíti a CurrentProjectFilePath változót a betöltött fájl elérési útjával, és meghívja az OnCurrentProjectChanged eseményt, hogy értesítse a rendszer többi részét a projektváltozásról.

A metódus minden lépése biztosítja, hogy a projekt pontos állapota a fájlból történő betöltés után teljesen helyreálljon, beleértve az oldalak forgatásának kezelését és a gyermekelemek megfelelő hozzárendelését.

A képen szöveg, képernyőkép, dokumentum, Betűtípus látható

Automatikusan generált leírás34. *ábra: A LoadProject metódus egy részlete.*

A SaveState metódus a projekt aktuális állapotát menti el az undo/visszavonás és redo/újra végrehajtás vermébe. A metódus először ellenőrzi, hogy a CurrentProject objektum és annak Document tulajdonsága érvényesek-e. Ha bármelyik értéke null, akkor a metódus nem végez semmilyen műveletet, és visszatér.

Amennyiben a projekt és a dokumentum érvényesek, a metódus a SerializeDocument segítségével az aktuális dokumentum állapotát XML formátumban tárolja el. Az így kapott dokumentumállapotot a Push metódus segítségével elmenti az \_undoStack verembe. Ez lehetővé teszi, hogy a felhasználó később visszavonja a módosításokat, ha szükséges.

A metódus ezt követően törli a \_redoStack veremet, mivel a redo verem akkor érvényes, ha új művelet történt, amely újra végrehajtható lenne.

A metódus biztosítja a megfelelő állapotkezelést a felhasználói műveletek során, lehetővé téve a projekt állapotának visszavonását, amíg új műveletek nem történnek.

Az Undo metódus lehetővé teszi a felhasználó számára, hogy visszavonja az utolsó módosítást, és visszaállítja a projekt dokumentumának korábbi állapotát. A metódus először ellenőrzi, hogy van-e elérhető állapot az \_undoStack veremben. Ha a verem üres, akkor nem történik semmilyen művelet, és a metódus visszatér.

Ha van elérhető állapot, a metódus első lépésként törli a jelenleg kiválasztott elemet az \_selectedElementStore.Select(null) hívással, biztosítva, hogy ne maradjon kijelölt elem a visszavonás alatt.

Ezután a metódus a SerializeDocument segítségével az aktuális dokumentum állapotát XML formátumban tárolja, és elmenti azt az \_redoStack verembe, hogy az újra végrehajtható legyen, ha a felhasználó később újra szeretné alkalmazni a műveletet.

A legutolsó állapotot a \_undoStack.Pop() hívásával eltávolítja a veremből, majd deszerializálja azt a DeserializeDocument metódussal. Ha a deszerializálás során hiba történik, vagy a visszaállított dokumentum null, akkor a metódus nem folytatódik.

A deszerializált dokumentum oldalait összehasonlítja az aktuális projekt dokumentum oldalainak tartalmával, és frissíti azok háttérszínét, valamint az oldal gyermekelemeit. Az oldalak gyermekelemeit a deszerializált állapotból másolja vissza a jelenlegi dokumentumba.

Végül a metódus meghívja az OnCurrentProjectChanged eseményt, hogy értesítse a rendszert a projekt változásáról, majd az AddAdorners metódus segítségével visszaállítja az képkereteket a projekt dokumentumához.

A képen szöveg, képernyőkép, Betűtípus, dokumentum látható

Automatikusan generált leírás35. *ábra: Az Undo metódus egy részlete.*

A Redo metódus lehetővé teszi a felhasználó számára, hogy újra végrehajtsa az utolsó visszavont műveletet, visszaállítva a projekt dokumentumának korábbi állapotát. A metódus először ellenőrzi, hogy van-e elérhető állapot az \_redoStack veremben. Ha a verem üres, a metódus nem hajt végre semmilyen műveletet, és visszatér.

Ha van elérhető állapot, a metódus első lépésként törli a jelenleg kiválasztott elemet az \_selectedElementStore.Select(null) hívással, biztosítva, hogy ne maradjon kijelölt elem a visszavonás vagy újra végrehajtás alatt.

Ezután az aktuális dokumentum állapotát a SerializeDocument segítségével XML formátumban tárolja, és elmenti azt az \_undoStack verembe, hogy a művelet később ismét visszavonható legyen, ha szükséges.

A következő állapotot a \_redoStack.Pop() hívásával eltávolítja a veremből, majd deszerializálja azt a DeserializeDocument metódussal. Ha a deszerializálás során hiba történik, vagy a visszaállított dokumentum null, akkor a metódus nem folytatódik.

A deszerializált dokumentum oldalait összehasonlítja az aktuális projekt dokumentum oldalainak tartalmával, és frissíti azok háttérszínét, valamint az oldal gyermekelemeit. Az oldalak gyermekelemeit a deszerializált állapotból másolja vissza a jelenlegi dokumentumba.

A metódus végül meghívja az OnCurrentProjectChanged eseményt, hogy értesítse a rendszert a projekt változásáról, majd az AddAdorners metódus segítségével visszaállítja az adornereket a projekt dokumentumához.

Végül a ProjectStore tartalmaz egy ClearUndoRedoStacks metódust, melyen keresztül ki lehet üríteni az undo és redo vermeket, ezáltal biztosítva, hogy ezek üresek legyenek egy nem visszavonható művelet elvégzését követően.

## 5.4 ResizeAdorner

A ResizeAdorner osztály a WPF Adorner leszármazottja, amely a méretezéshez használt fogópontokat (thumb) biztosítja egy adott UI elemhez, lehetővé téve annak átméretezését a felhasználó számára. Az osztály különböző eseményeket kezel, mint például a DragStarted, DragDelta, és DragCompleted, hogy az elemet megfelelően átméretezze és a szükséges műveleteket végrehajtsa.

A konstruktorban a következő elemek kerülnek létrehozásra:

* thumbBounds: egy Rectangle, amely a méretezési határokat jeleníti meg a felhasználó számára.
* thumbTopLeft, thumbTopRight, thumbBottomLeft, thumbBottomRight: az egyes sarkokhoz rendelt Thumb objektumok, amelyek lehetővé teszik az elem átméretezését az adott sarkok mentén. Mindegyik thumb különböző kurzorokkal rendelkezik, amelyek az adott irányú méretezést jelzik.

A méretezési műveletek során a Thumb\_DragDelta eseménykezelő az átvitt elmozdulásokat (HorizontalChange és VerticalChange) felhasználva módosítja az elem szélességét és magasságát, figyelembe véve az esetleges forgatásokat és skálázásokat, hogy az elem a kívánt módon méreteződjön.

A Thumb\_DragStarted esemény előtt elmentésre kerül a projekt állapota a \_projectStore.SaveState() metódussal, hogy biztosítsuk az esetleges visszavonási lehetőségeket.

A Thumb\_DragCompleted esemény végrehajtása után az \_isStateSaved változó visszaállításra kerül, jelezve, hogy az átméretezés befejeződött.

A képen szöveg, képernyőkép, dokumentum, Betűtípus látható

Automatikusan generált leírás A VisualChildrenCount és a GetVisualChild metódusok biztosítják a vizuális elemek kezelését, így a méretezési kezelők is megjeleníthetők a képernyőn. Az ArrangeOverride metódus a végleges elrendezést biztosítja, míg a MeasureOverride metódus a gyermekelemek mérését végzi el a rendelkezésre álló hely függvényében.

36. *ábra: A Thumb\_DragDelta metódus egy részlete.*

## 5.5 FrameAdorner

A FrameAdorner osztály szintén a WPF Adorner leszármazottja, amely egy képet jelenít meg egy adott UI elem körül, így keretet adva annak. A képet az ImageSource tulajdonság tárolja, amelyet a konstruktoron keresztül adhatunk meg. Az osztály célja, hogy egy vizuális keretet helyezzen el az elemen, ezzel megvalósítva a fotókönyv képkeret funkcióját.

A konstruktorban létrehozásra kerül egy Image objektum, amely a megadott ImageSource értéket használja a képként való megjelenítéshez. Az Image objektum IsHitTestVisible tulajdonsága hamisra van állítva, biztosítva, hogy a keret ne legyen interaktív, vagyis ne reagáljon a felhasználói eseményekre, például kattintásra. Ezen kívül a Stretch tulajdonságot Stretch.Fill értékre állítjuk, így a kép kitölti a rendelkezésre álló teret, és StretchDirection.Both értékkel biztosítjuk, hogy a kép mind vízszintesen, mind függőlegesen legyen nyújtva, ha szükséges.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás A VisualChildrenCount és GetVisualChild metódusok biztosítják a képi elemek kezelését, így a keret megjeleníthető a képernyőn. Az ArrangeOverride metódus a végleges elrendezést végzi el, ahol a képet úgy helyezi el, hogy az 12%-kal túlterjedjen az eredeti elem szélein, ezáltal körbefogva azt. A MeasureOverride metódus a gyermekelemek mérését végzi el a rendelkezésre álló hely függvényében.

37. *ábra: A FrameAdorner osztály egy részlete.*

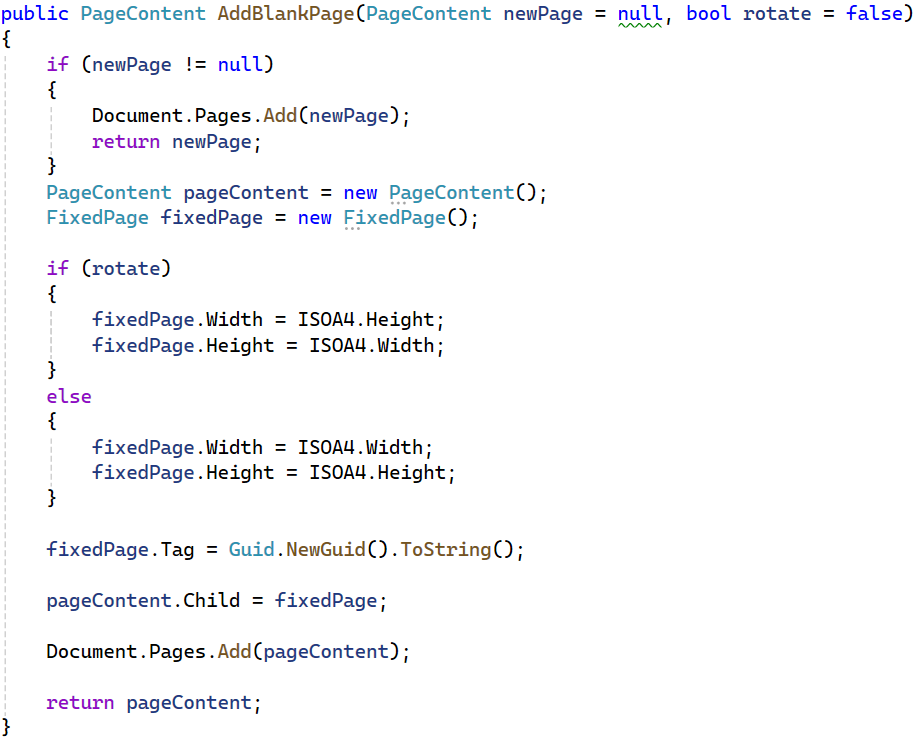
## 5.6 Project.cs

A Project osztály a projekt dokumentumot és annak kezelését valósítja meg. Felelős a projekt oldalainak és tartalmának létrehozásáért, módosításáért és a hozzáadott elemek kezeléséért. A projekt dokumentuma FixedDocument típusú, amely lehetővé teszi a különböző típusú elemek, például szövegek és képek elhelyezését a dokumentumban.

A konstruktorban létrejön egy üres dokumentum, amely az alapértelmezett ISO A4-es méretű oldalakkal van inicializálva. Az AddBlankPage metódus segítségével új üres oldalakat adhatunk hozzá a dokumentumhoz, és az oldalakat elforgathatjuk, ha szükséges. Az oldalakat PageContent objektumokként kezeljük, amelyek FixedPage típusú gyermekeit tartalmazzák. Az AddTestElements metódus a debug módban történő futtatás esetén alapértelmezett elemeket ad hozzá, ezzel elősegítve a manuális tesztelést.

A szövegelemeket TextBlock objektumokkal valósítjuk meg, amelyeket a GetTextField metódus generál. A szövegekhez tartozó kontextus menü segítségével a felhasználók törölhetik az elemeket. A képelemeket Image objektumok segítségével hozzuk létre, melyekhez szintén kontextus menü tartozik, lehetőséget biztosítva a képek módosítására vagy eltávolítására, és a képkereteik törlésére.

A SetActivePage metódussal beállíthatjuk az aktuálisan aktív oldalt, amely a dokumentum manipulálásához szükséges. Az ActivePageChanged esemény értesíti a megfelelő nézetmodelleket, ha az aktív oldal változik.

 A projekt dokumentumának kezeléséért a Dispose metódus felel, amely felszabadítja az erőforrásokat, és eltávolítja a kiválasztott elemet, valamint az aktív oldalt.

38. *ábra: A Project osztály AddBlankPage metódusa.*

## 5.7 Főbb parancsok

Az alkalmazás különböző funkcióit úgynevezett parancsokkal implementáljuk, melyeket bizonyos felhasználói eseményekhez kötünk, mint például egy gombnyomás. Ezeket a parancsokat a CommandBase osztályból származtatjuk.

A CommandBase osztály az ICommand interfész absztrakt implementációját valósítja meg. A CanExecute és Execute metódusok implementálása révén lehetővé teszi a parancsok végrehajtását és azok engedélyezésének ellenőrzését. Az Execute metódust az éppen adott parancs osztályában írjuk felül, és itt adhatjuk meg a tényleges funkcionalitást.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás39. *ábra: A CommandBase osztály.*

### 5.7.1 Varázsló

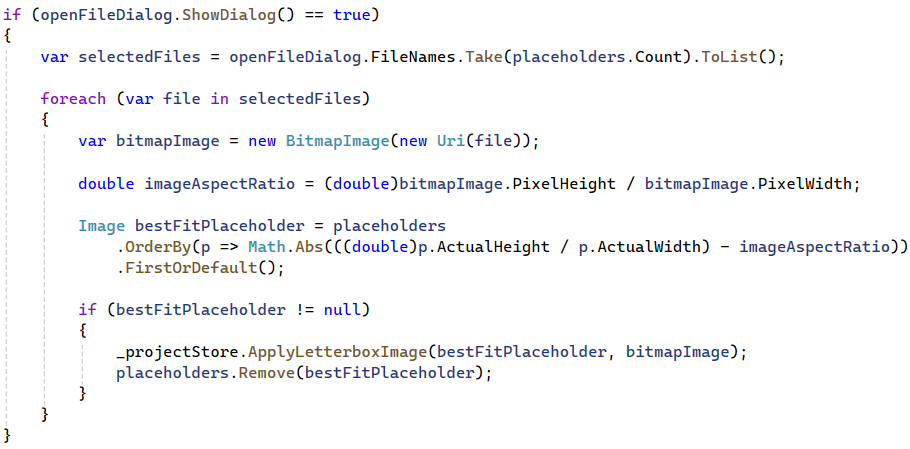
A RunWizardCommand parancs felelős a varázsló folyamat elindításáért, amely képes képeket hozzáadni a projekt dokumentumához, és azokat a dokumentum oldalainak háttérszínéhez illeszkedő színre állítja. Az osztály a ProjectStore példányát használja a projekt adatainak eléréséhez, és a projekt állapotát is a műveletek előtt.

Az Execute metódusban első lépésként ellenőrizzük, hogy a projekt dokumentuma létezik-e, és ha nem, akkor visszatérünk. Ezután a varázsló végigiterál a dokumentum oldalain, és megkeresi azokat az Image típusú elemeket, amelyek a "placeholder" szót tartalmazzák a kép forrásában. Ha talál ilyen helyettesítő képeket, akkor egy fájlkezelő ablak ugrik fel, ahol a felhasználó képeket választhat ki, amelyek az adott helyettesítő képeket váltják fel.

A legmegfelelőbb képet a kép arányának összehasonlításával választja ki, hogy az optimálisan illeszkedjen a helyettesítő kép méretéhez.

Ezt követően a program minden egyes oldalhoz hozzáadja a képeket, és kiszámítja a képek átlagos színét a háttérszínhez való illesztés céljából. A képek RGB komponenseit összegyűjtve határozza meg a háttérszínt.

Ezután a GetComplementaryColor metódus az aktuális háttérszín komplementer színét számítja ki, amelyet aztán a szövegek színének beállításához használ.

40. *ábra: A RunWizardCommand egy részlete.*

### 5.7.2 Színválasztó

A ChangeColorCommand parancs felelős a színek megváltoztatásáért a projekt dokumentumában. A parancs a projekt állapotának mentésével indítja el a színváltoztatást, majd ellenőrzi a paraméter típusát, hogy biztosítsa a megfelelő művelet végrehajtását. Az osztály a ProjectStore és a SelectedElementStore példányait használja annak érdekében, hogy elérje az aktuális projektet és az aktuálisan kiválasztott elemet.

Az Execute metódus végzi el a színcserét. Elsőként ellenőrzi, hogy a paraméter valóban egy Button típusú objektum-e, és ha nem, a metódus visszatér anélkül, hogy bármit is végrehajtana. Ezután ellenőrzi, hogy a projekt dokumentuma elérhető-e. Ha ezek a feltételek teljesülnek, akkor tovább vizsgálja az aktív oldalt és a kiválasztott elemet.

Amennyiben a kiválasztott elem egy TextBlock, úgy a TextBlock szövegszínét a gomb háttérszínére állítja. Ha nem egy szöveges blokk, de az aktív oldal rendelkezésre áll, akkor az oldal háttérszínét módosítja a gomb háttérszínére.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás41. *ábra: A ChangeColorCommand Execute metódusa.*

### 5.7.3 Képkeretek, szövegbuborékok

A képkeretek és szövegbuborékok felugró ablakkal történő hozzáadásáért az OkCollectionModalCommand parancs felel. A konstruktorban megkapja a szükséges nézetmodellt, navigációs szolgáltatást, a projektet tároló osztályt és egy opcionális SelectedElementStore példányt.

A Execute metódus felelős az alapvető logikáért, amely először ellenőrzi, hogy a kiválasztott kép létezik-e. Ha nem található, a művelet nem hajtódik végre. Amennyiben a SelectedElementStore nincs megadva, az aktuális projekthez hozzáadja az új szövegbuborékot. Ha a SelectedElementStore elérhető, és egy képelem van kiválasztva, akkor a parancs végrehajtása előtt ellenőrzi, hogy az elem megfelelő-e a módosításhoz (pl. nem rendelkezik egy "ignore\_wizard" címkével, vagyis nem szövegbuborék). Ezt követően a projekt állapota mentésre kerül, a régi képkeret eltávolítódik, majd az új FrameAdorner-t hozzáadjuk a kiválasztott elemhez.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás42. *ábra: Az OkCollectionModalCommand Execute metódusa.*

### 5.7.4 Sablonok

A sablonok felugró ablakkal történő hozzáadásáért az OkTemplateModalCommand parancs felelős. A konstruktorban a sablonok kezeléséért felelős TemplateBrowserViewModel példány kerül átadásra, amely lehetővé teszi a sablonok betöltését és alkalmazását a projektben.

A Execute metódusban a parancs meghívja a AddPage metódust, amely a sablonok hozzáadását végzi el. Ez a metódus egy új oldalt hoz létre a projektben, és betölti a megfelelő sablont, ha az aktuális elem a "custom\_template" típusú. Ha a sablon nem "custom\_template" típusú, akkor egyszerűen a kiválasztott elem gyermekelemeit másolja az új oldalra.

Az AddPage metódus a következő lépéseket hajtja végre:

* Ha a sablon "custom\_template" típusú, akkor egy fájlmegnyitó párbeszédpanel jelenik meg, amely lehetővé teszi a felhasználó számára, hogy kiválassza a sablonfájlt. A fájlt XML formátumban olvassa be, majd deszerializálja azt egy FixedPage objektummá.
* Ha a sablon sikeresen betöltődött, a sablon tartalma (gyermekelemek) átkerül az új oldalra.
* A gyermekelemeket új GUID-okkal látja el, és hozzáadja őket az új oldalhoz.
* A projekt történetét törli (undo/redo veremek), mivel ez a művelet a FixedDocument kezelésével kapcsolatos problémák miatt különböző anomáliákat okozna visszavonás esetében.

Miután az oldal hozzáadása befejeződött, a navigációs szolgáltatás segítségével a rendszer navigál a következő nézetre, hogy a felhasználó láthassa a módosított dokumentumot.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás43. *ábra: Az AddPage metódus egy részlete.*

### 5.7.5 Kijelölt oldal törlése

A kijelölt oldal törléséért a DeletePageCommand parancs felel. Lehetővé teszi a dokumentumban található aktív oldal törlését. A konstruktorban két osztálypéldány kerül átadásra: a SelectedElementStore, amely a kiválasztott elemek nyomon követéséért felel, és a ProjectStore, amely a projektet és annak oldalait kezeli.

Az Execute metódusban a parancs végrehajtja a következő műveleteket:

* Ellenőrzi, hogy a projekt és az aktív oldal létezik-e, valamint, hogy a dokumentumban több mint egy oldal található-e. Ha bármelyik feltétel nem teljesül, a művelet nem hajtódik végre.
* A felhasználó számára egy megerősítő üzenet jelenik meg, amely figyelmezteti a törlés visszavonhatatlanságára. Ha a felhasználó nem erősíti meg a törlést, a művelet megszakad.
* Ha a felhasználó megerősíti a törlést, a parancs először egy új projektet hoz létre, amelynek oldalaiban az összes oldalt újra létrehozza, kivéve a törlésre szánt oldalt. Az oldalak hátterét, címkéjét és gyermekelemeit átmásolja az új projektbe.
* Miután a törlés megtörtént, az új projektet beállítja a ProjectStore.CurrentProject változóban, így a projekt frissül és az új dokumentumot tartalmazza.
* Az oldalak új példányai a megfelelő adatokkal (háttér, címke, gyermekelemek) kerülnek átmásolásra.
* A művelet végén törli az undo/redo vermeket, így biztosítva, hogy a törlés után ne legyenek érvényben a visszavonási műveletek.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás44. *ábra: Az OkCollectionModalCommand Execute metódusának egy részlete.*

### 5.7.6 Kijelölt oldal elforgatása

A RotatePageCommand parancs a kijelölt oldal elforgatását valósítja meg. A konstruktorban két osztálypéldány kerül átadásra: a SelectedElementStore, amely a kiválasztott elemek kezeléséért felelős, és a ProjectStore, amely a projekt oldalait és azok adatait tárolja.

A Execute metódusban a következő lépések hajtódnak végre:

* Ellenőrzi, hogy a projekt és az aktív oldal létezik-e. Ha bármelyik feltétel nem teljesül, a művelet nem hajtódik végre.
* A pageToRotate változóban tárolja az aktív oldalt, és meghatározza annak indexét a dokumentumban.
* A rendszer az összes már elforgatott oldalt összegyűjti, hogy figyelembe vegye őket a további műveletek során. Az elforgatott oldalak azok, amelyek szélessége nagyobb, mint a magasságuk.
* Egy új Project példányt hoz létre, amely biztosítja, hogy az oldal elforgatása után a projekt struktúrája frissüljön.
* Az oldalak átmásolásakor figyelembe veszi a forgatás típusát:
  + Ha az oldal függőleges (magasság nagyobb, mint a szélesség), akkor az új projektbe új üres oldalakat ad hozzá, figyelembe véve az elforgatott oldalakat.
  + Ha az oldal vízszintes (szélesség nagyobb, mint a magasság), akkor az új oldal elforgatásra kerül.
* Az oldalon található gyermekelemek helyzete is frissül a forgatás hatására. Az elemek bal és felső pozícióját megcseréli, hogy az új elrendezésnek megfelelően jelenjenek meg a dokumentumban.
* Az új oldalak hozzáadása után az új projektet beállítja a ProjectStore.CurrentProject változóba, és minden oldal tartalmát átmásolja az új projektbe.
* Az undo/redo vermeket törli, hogy a forgatás után ne maradjanak érvényben a visszavonási műveletek.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás45. *ábra: A RotatePageCommand egy részlete.*

### 5.7.7 Kijelölt oldal mentése sablonként

A SavePageAsTemplateCommand parancs lehetővé teszi a felhasználó számára, hogy az aktív oldalt sablonként elmentse. A parancs a ProjectStore példányon keresztül fér hozzá a projekthez és annak tartalmához.

A Execute metódusban a következő lépések történnek:

* Előfeltételek ellenőrzése: Ellenőrzi, hogy a projekt, a dokumentum és az aktív oldal létezik-e. Ha bármelyik nem elérhető, a művelet nem hajtódik végre.
* Oldal előkészítése a mentéshez:
  + A kiválasztott oldal (FixedPage) gyermekelemeinek méretét beállítja, ha azok nem meghatározottak (Width és Height) Ez szükséges, mivel ha még nem méreteztünk át egy adott elemet, annak a Width és Height tulajdonsága NaN-t ad vissza.
  + Az oldal adatait (beleértve a gyermekelemeket) szöveges formátumba (XML) szerializálja a SerializeFixedPage metódus segítségével.
* Fájl mentés:
  + A felhasználó számára megjelenik egy fájltárolási párbeszédpanel (SaveFileDialog), amely lehetővé teszi egy .template fájl mentését. Ha a felhasználó válaszol és megadja a fájl elérési útját, akkor a szerializált adatokat a megadott fájlba menti.
* Kontextusmenük kezelése:
  + A mentés előtt és után eltávolítja, illetve újra létrehozza a gyermekelemek kontextusmenüit (ContextMenu), mivel ezeket a szerializáló metódus nem képes szerializálni.

A képen szöveg, képernyőkép, Betűtípus, dokumentum látható

Automatikusan generált leírás46. *ábra: A SavePageAsTemplateCommand Execute metódusának egy részlete.*

## 5.8 Képszerkesztési műveletek

### 5.8.1 Elemek forgatása

A kódban a RotationDegree tulajdonság kezeli az elemek forgatását. A RotationDegree getter és setter segítségével lehet lekérdezni és beállítani a forgatás mértékét.

A RotationDegree getterében először ellenőrizzük, hogy az elem RenderTransform tulajdonsága egy TransformGroup típusú objektum-e. Ha igen, akkor megkeressük a benne található RotateTransform-ot, és visszaadjuk annak szögét (Angle). Ha nem található RotateTransform, akkor 0 fokot adunk vissza.

A setterben először mentjük az aktuális állapotot a \_projectStore.SaveState() hívásával, hogy később vissza lehessen állítani, ha szükséges. Ezután korlátozzuk a szöget -360 és 360 fok között. Ha a szög túl nagy vagy túl kicsi, akkor normalizáljuk azt.

Ha a RenderTransform nem tartalmaz TransformGroup-ot, akkor létrehozunk egy új TransformGroup-ot, majd hozzáadunk egy RotateTransform-ot, ha még nincs benne. A RotateTransform beállítja a forgatás középpontját (általában az elem közepét, azaz a CenterX és CenterY értékeit), majd beállítja a kívánt forgatási szöget.

Végül az OnPropertyChanged eseményt hívjuk, hogy értesítsük a UI-t a tulajdonság változásáról, és így a felhasználói felület frissül.

### 5.8.2 Elemek tükrözése

Az elemek tükrözését a HorizontalMirrorCommand és VerticalMirrorCommand parancsok valósítják meg.

Az Execute metódus elején a projekt állapota elmentésre kerül a \_projectStore.SaveState() hívásával, hogy a felhasználó később vissza tudja állítani a módosításokat, ha szükséges.

A tükrözéshez a RenderTransform tulajdonságot használjuk. Ez a tulajdonság lehetővé teszi az elem vizuális átalakítását anélkül, hogy a tényleges elhelyezkedése változna a layoutban.

Először ellenőrizzük, hogy az elem RenderTransform tulajdonsága már tartalmaz-e egy TransformGroup objektumot. Ha nem, akkor létrehozunk egyet, és hozzárendeljük az elemhez.

A ScaleTransform az egyik típusú átalakítási objektum, amely a skálázást végzi. Ellenőrizzük, hogy a TransformGroup tartalmaz-e már ScaleTransform-ot. Ha nem, létrehozunk egy újat, és hozzáadjuk a TransformGroup gyermekeinek.

A ScaleTransform középpontjának beállítása fontos, hogy az átalakítás az elem közepére történjen. Ehhez a középpontot az elem szélességének és magasságának felére állítjuk be.

Ezután, hogy tükrözzük az elemet, a ScaleY értékét negatívra állítjuk. A negatív ScaleY érték a vízszintes tükrözést eredményezi, míg a ScaleX értékét nem módosítjuk, így az elem nem fog függőlegesen elforogni. Ezzel megegyező módon függőleges tükrözés esetében a ScaleX tulajdonságot módosítjuk.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás47. *ábra: A HorizontalMirrorCommand Execute metódusa.*

### 5.8.3 Elemek pozícionálása

A pozicionálás két fontos tulajdonságot használ a PropertiesPanelViewModel osztályban: ElementX és ElementY. Ezek a tulajdonságok határozzák meg az elem vízszintes (X) és függőleges (Y) elhelyezkedését az oldalon.

ElementX:

* Az elem vízszintes elhelyezkedését a FixedPage.GetLeft metódus segítségével számítjuk ki. Ha az érték nem érvényes (double.IsNaN), akkor az \_elementX változóra alapozva történik a visszatérés. Ha a szülő elem egy Image, akkor annak a szülőjét vesszük alapul. A pozíció nem léphet túl meghatározott határokon: fekvő tájolás esetén maximum 1124, álló esetén 797 pixelre van korlátozva.

ElementY:

* Hasonlóan a ElementX-hez, az elem függőleges pozícióját a FixedPage.GetTop metódus határozza meg. Az érték szintén korlátozva van, hogy az elem ne lépje túl a maximális elhelyezkedést az oldalon (fekvő tájolás esetén 797, álló esetén 1124 pixel).

A pozíció változtatásakor az új értéket a megfelelő FixedPage.SetLeft és FixedPage.SetTop metódusokkal alkalmazzuk, ezáltal az elem új helyre kerül.

### 5.8.4 Elemek méretezése

A méretezést a ResizeWidth és ResizeHeight tulajdonságok kezelik, amelyek lehetővé teszik az elem szélességének és magasságának módosítását. A ResizeWidth és ResizeHeight mindig az aktuális elem szélességét és magasságát tükrözik, amelyeket a felhasználó állíthat be.

ResizeWidth:

* A ResizeWidth tulajdonság módosításakor először ellenőrizzük, hogy az új érték ne legyen kisebb, mint 1. Ha a KeepAspectRatio (arány megtartása) értéke igaz, akkor a magasságot is automatikusan módosítjuk úgy, hogy megmaradjon a szélesség/magasság arány. Ha az arány nem szükséges, csak a szélességet változtatjuk meg, és a magasságot külön kell beállítani.

ResizeHeight:

* A ResizeHeight működése hasonló a szélességhez. Az értéke nem lehet kisebb, mint 1. Az arány megtartása esetén a szélesség automatikusan változik a magasság értékével együtt. Ha az arányt nem kell megtartani, akkor a magasságot egyedileg változtathatjuk, és külön kell kezelni a szélességet.

### 5.8.5 Rétegek kezelése

Az elemek fedésének kezelését a MoveElementBackwardCommand és MoveElementForwardCommand parancsok kezelik. Mielőtt bármilyen módosítást végeznének az elem pozícióján, a parancsok első lépése a projektállapot mentése a \_projectStore.SaveState() hívásával. Ez biztosítja, hogy az aktuális állapot el legyen tárolva, és szükség esetén visszaállítható legyen.

A következő lépés a kiválasztott elem szülőjének lekérdezése és annak ellenőrzése, hogy valóban egy olyan konténerben található-e, amely képes gyermek elemeket tartani. A kiválasztott elem szülőjét a \_selectedElement.Parent segítségével érjük el, és biztosítjuk, hogy az származik-e FrameworkElement típusú objektumból.

Ha a szülő elem AdornerDecorator, amely tipikusan egy vizuális díszítő elem, akkor az elem valódi szülőjét a parent.Parent lekérdezésével találjuk meg, mivel az adorner konténer egy külön szülő elemen keresztül tárolja a valódi gyermek elemeket.

Ha a szülő valóban FixedPage, akkor az elem eltávolításra kerül a szülő konténerből, majd újra beszúrásra kerül a FixedPage.Children.Insert(0, element) hívásával. Az Insert(0, element) hívás biztosítja, hogy az elem a lista első pozíciójába kerüljön, azaz a legalsó rétegre (hátrébb a többi elemhez képest). Fordított esetben a legutolsó helyre szúrjuk be az elemet. A rétegek kezelése azért ilyen módon történik, mert a FixedDocument nem kezel Z indexeket, így a dokumentum fa módosításával kell elérnünk, hogy az elemek megfelelően fedjék egymást.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás48. *ábra: A MoveElementBackwardCommand Execute metódusa.*

### 5.8.6 Szöveges mezők betűméretének, betűtípusának és betűstílusának kezelése

Ha a kijelölt elemünk egy szöveges mező akkor lehetőségünk van a betűméret, betűtípus, és betűstílus beállítására is.

A PropertiesPanelViewModel osztályban a SelectedFont tulajdonság felelős a betűtípus kiválasztásáért.

A felhasználó által választott betűtípus a SelectedFont tulajdonságban van tárolva, amely az elem típusától függően frissíti az adott szöveges elem betűtípusát.

Az elérhető betűtípusokat a rendszer összes telepített betűtípusából töltjük be, és rendezzük őket az AvailableFonts listába. Ez lehetővé teszi a felhasználó számára, hogy válasszon az elérhető betűtípusok közül.

Miután a felhasználó kiválasztotta a kívánt betűtípust, a SelectedFont tulajdonság beállítása automatikusan frissíti az adott szöveges elem TextElement.FontFamily tulajdonságát, így az új betűtípus alkalmazásra kerül.

A PropertiesPanelViewModel osztályban a FontSize tulajdonság felelős a szöveges elem betűméretének módosításáért.

A FontSize tulajdonság lehetővé teszi a felhasználó számára, hogy megadja a kívánt betűméretet. A TextElement.FontSize tulajdonságon keresztül a módosított betűméret automatikusan alkalmazásra kerül a szöveges elemre.

A betűméret beállítása előtt ellenőrizzük, hogy a megadott érték legalább 1, mivel a 0 vagy negatív értékek nem érvényesek. A szöveg vizuális frissítése érdekében a OnPropertyChanged(nameof(FontSize)) hívással értesítjük a nézetet a változtatásról.

A betűstílusok beállításához PropertiesPanelViewModel osztályban három parancs van definiálva.

Félkövér (Bold):

* A ToggleBoldCommand parancs alkalmazza a félkövér stílust a TextBlock elemre. A parancs végrehajtásakor a szöveg FontWeight tulajdonsága változik, és a szöveg félkövérré válik, ha a felhasználó aktiválja, illetve eltávolítja, ha deaktiválja.

Dőlt (Italic):

* A ToggleItalicCommand parancs a dőlt stílust alkalmazza a TextBlock elemre. A FontStyle tulajdonságot módosítja, így a szöveg dőlt stílusban jelenik meg, ha a felhasználó aktiválja.

Aláhúzott (Underline):

* Az ToggleUnderlineCommand parancs az aláhúzást alkalmazza a TextBlock elemre, módosítva annak TextDecorations tulajdonságát, és így a szöveg aláhúzott lesz, ha a felhasználó aktiválja.

A képen szöveg, Betűtípus, sor, képernyőkép látható

Automatikusan generált leírás49. *ábra: A ToggleUnderlineCommand Execute metódusa.*

# 6. Összegzés

A projektünk keretében egy funkciókban gazdag, esztétikus felhasználói felülettel rendelkező fotókönyv szerkesztő alkalmazást fejlesztettünk. Sikerült elérni minden kitűzött célunkat, és ezen felül is extra funkciókkal tudtuk bővíteni alkalmazásunkat. Összességében elégedettek vagyunk a szoftverrel, egy meglehetősen piacképes, felhasználóbarát alkalmazást sikerült fejlesztenünk. A fejlesztés során számos új technológiával ismerkedtünk meg, mint például az MVVM, vagy a dependency injection. A csapatmunka készsége is sokat fejlődött a projekt készítése során, és hasznos ismeretekkel bővült a tudásunk, melyeket a jövőben könnyedén alkalmazhatunk majd. Továbbfejlesztés szempontjából figyelembe lehetne venni bizonyos szempontokat, mint például a fotókönyvek automatikus elkészítésének további javítása. Alkalmazásunk jelenleg is rendelkezik funkciókkal, melyek ezt segítik elő. Ilyen például a varázsló, vagy a sablonok. Ezen felül viszont még tovább lehetne fejleszteni ezeket, ezáltal még felhasználóbarátabbá téve az alkalmazást, és megkönnyíteni a kevésbé hozzáértő felhasználók munkáját. Ilyen funkció lehetne például egy AI alapú segéd, amely sokkal személyre szabottabb ajánlásokat tenne, a fotókönyv témája és képei alapján. Egy ilyen funkció sokkal piacképesebbé tenné szoftverünket, mivel napjainkban az AI használata rengeteg területen megjelenik, és „trendivé” teszi a manapság elérhető alkalmazásokat. Piackutatásunk során is csupán egyetlen alkalmazással találkoztunk, mely rendelkezett AI funkciókkal, így mindenképpen ez lehetne a legfőbb továbbfejlesztési irány. Összességében mindannyian elégedettek vagyunk a fejlesztéssel és a csapatmunkánkkal.

# Irodalomjegyzék

[1] <https://www.vikingsoftware.com/wp-content/uploads/2024/02/C.png> (Megtekintés dátuma: 2024. 11. 27.)

[2] <https://journaldev.nyc3.cdn.digitaloceanspaces.com/2018/04/android-mvvm-pattern.png> (Megtekintés dátuma: 2024. 11. 27.)

[3] <https://support.journiapp.com/hc/article_attachments/21086244161565> (Megtekintés dátuma: 2024. 11. 27.)

[4] <https://cewe.hu/> (Megtekintés dátuma: 2024. 11. 27.)

[5] <https://en.internetprint.eu/> (Megtekintés dátuma: 2024. 11. 27.)

[6] <https://myprintpix.hu/> (Megtekintés dátuma: 2024. 11. 27.)

[7] <https://www.sooters.hu/> (Megtekintés dátuma: 2024. 11. 27.)

[8] <https://www.prodigi.com/> (Megtekintés dátuma: 2024. 11. 27.)

[9] [https://content-management-files.canva.com/cdn-cgi/image/f=auto,q=70/4847789c-c1e1- 4bdf-a43a-0ead490284dc/photobook\_how-to2x.png](https://content-management-files.canva.com/cdn-cgi/image/f=auto,q=70/4847789c-c1e1-%094bdf-a43a-0ead490284dc/photobook_how-to2x.png) (Megtekintés dátuma: 2024. 11. 27.)

[10] <https://www.javatpoint.com/csharp-features> (Megtekintés dátuma: 2024. 11. 27.)

[11] <https://www.javatpoint.com/c-sharp-object-and-class> (Megtekintés dátuma: 2024. 11. 27.)

[12] [https://learn.microsoft.com/en-us/visualstudio/debugger/debugger-feature-tour?view=vs- 022](https://learn.microsoft.com/en-us/visualstudio/debugger/debugger-feature-tour?view=vs-%09022) (Megtekintés dátuma: 2024. 11. 27.)

[13] <https://learn.microsoft.com/en-us/visualstudio/ide/using-intellisense?view=vs-2022> (Megtekintés dátuma: 2024. 11. 27.)

[14] <https://visualstudio.microsoft.com/services/live-share/> (Megtekintés dátuma: 2024. 11. 27.)

[15] <https://www.javatpoint.com/net-framework> (Megtekintés dátuma: 2024. 11. 27.)

[16] <https://learn.microsoft.com/en-us/dotnet/desktop/wpf/overview/?view=netdesktop-7.0> (Megtekintés dátuma: 2024. 11. 27.)

[17] <https://learn.microsoft.com/en-us/dotnet/desktop/wpf/data/?view=netdesktop-7.0> (Megtekintés dátuma: 2024. 11. 27.)

[18] <https://learn.microsoft.com/en-us/dotnet/desktop/wpf/xaml/?view=netdesktop-7.0> (Megtekintés dátuma: 2024. 11. 27.)

[19] <https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm> (Megtekintés dátuma: 2024. 11. 27.)

[20] <https://learn.microsoft.com/en-us/dotnet/architecture/maui/dependency-injection> (Megtekintés dátuma: 2024. 11. 27.)

[21] <https://docs.github.com/en/get-started/start-your-journey/about-github-and-git> (Megtekintés dátuma: 2024. 11. 27.)

[22] <https://www.journiapp.com/> (Megtekintés dátuma: 2024. 11. 27.)

[23] <https://www.canva.com/photo-books/> (Megtekintés dátuma: 2024. 11. 27.)