UNIVERZITA J. SELYEHO – SELYE JÁNOS EGYETEM

FAKULTA EKONÓMIE A INFORMATIKY –
GAZDASÁGTUDOMÁNYI ÉS INFORMATIKAI KAR

DIGITÁLNY FOTOALBUM -
DIGITÁLIS FOTÓALBUM

Bakalárska práca – Szakdolgozat

Dávid Demeter

2022

UNIVERZITA J. SELYEHO – SELYE JÁNOS EGYETEM

FAKULTA EKONÓMIE A INFORMATIKY –
GAZDASÁGTUDOMÁNYI ÉS INFORMATIKAI KAR

DIGITÁLNY FOTOALBUM -
DIGITÁLIS FOTÓALBUM

Bakalárska práca – Szakdolgozat

Študijný program:	Aplikovaná informatika
Tanulmányi program:	Alkalmazott informatika
Názov študijného odboru:	18. informatika
Tanulmányi szak megnevezése:	18. informatika
Vedúci záverečnej práce:	PaedDr. Krisztina Czakóová, PhD.
Témavezető:	PaedDr. Czakóová Krisztina, PhD.
Školiace pracovisko:	Katedra informatiky
Tanszék megnevezése:	Informatikai Tanszék

Dávid Demeter

Komárno, 2022

Univerzita J. Selyeho
Fakulta ekonómie a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Dávid Demeter
Študijný program: Aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 18. - informatika
Typ záverečnej práce: Bakalárska práca
Jazyk záverečnej práce: maďarský
Sekundárny jazyk: slovenský
Téma: Digitális fotóalbum
Anotácia: A záródolgozat célja egy olyan digitális fotóalbum létrehozása, mely a feltöltött fotókat egy kiválasztott oldalszámú virtuális könyvszerű fotóalbum felületre engedi elhelyezni (kiválasztás után), majd a fotókat címmel/kommenttel is ellátni, igény esetén. Az elhelyezett fotókat képes forgatni, méretezni tetszés szerint a lapon. A fotók hátterébe választhatók különböző témák (keretek, minták, stílusok, rajzok, stb.). A végeredmény egy pdf fájlban kerül mentésre (nyomtatásra).
Kľúčové slová: digitális képfeldolgozás, képkezelés, háttér téma beállítása, fájlba mentés

Vedúci: PaedDr. Krisztina Czakóová, PhD.
Katedra: KINF - Katedra informatiky
Vedúci katedry: Ing. Ondrej Takáč, PhD.

Dátum schválenia: 02.09.2021

prof. Dr. Annamária Várkonyiné Kóczy, DSc.
garant študijného programu

Čestné vyhlásenie – Becsületbeli nyilatkozat

Čestne vyhlasujem, že bakalársku prácu som vypracoval samostatne s použitím uvedenej literatúry a pod odborným vedením vedúcej práce.

Kijelentem, hogy a szakdolgozatot önállóan dolgoztam ki a feltüntetett irodalom felhasználásával és témavezetőm szakmai irányításával.

Komárno, 2022

Komárom, 2022 .....

vlastnoručný podpis – sajátkezű aláírás

Köszönetnyílvánítás

Ezúton köszönetet szeretnék mondani a témavezetőmnek PaedDr. Czakóová Krisztina, PhD. tanárnőnek a segítségért, mivel hasznos tanácsokkal és észrevételekkel látott el a munkám során.

Továbbá köszönet illeti a családomat és mindenki mást is, akik valamilyen formában támogattak.

Abstrakt

Cieľom tejto bakalárskej práce je vytvorenie digitálneho fotoalbumu, kde je možné nahrané obrázky po výbere umiestniť na plochu podobnej knihe, ľubovoľne ich označiť popisom (textom), umiestnené obrázky otáčať a meniť ich veľkosť na pozadí. Možnosťou je nastavenie pozadia, rámu, štýlu ako aj kresieb a odtlačkov rôzných vzoriek. Nakoniec je možnosť album uložiť v pdf formáte. Práca sa skladá z piatich kapitol, 5 obrázkov a z CD prílohy. V prvých dvoch v kapitolách sme sa zamerali na fotoalbum a trhový prieskum. Tretia a štvrtá kapitola opisuje techniku a presné podrobnosti vyhotovenia digitálneho fotoalbumu. V piatej kapitole sme zosumarizovali výsledky a predkladáme návrhy na ďalší vývoj.

Kľúčové slová: digitálne spracovanie obrazu, správa obrazu, nastavenie témy pozadia, uloženie do súboru

Absztrakt

A szakdolgozat célja egy digitális fotóalbum létrehozása, ahol kiválasztás után elhelyezhetők a feltöltött képek egy könyvszerű felületen, tetszés szerint feliratokkal ellátható, az elhelyezett képek forgathatók, méretezhetők a lapon. Hátterek, keretek, stílusok, rajzok, minták is választhatók, igény szerint. Legvégül elmenthető pdf formátumban. A munka 5 fő fejezetre épül, 5 ábrát és 1 CD mellékletet tartalmaz. Az első két fejezetben ismertetjük a célunkat az albummal kapcsolatban, valamint a piacon lévő alternatívákat mutattuk be. A harmadik és negyedik fejezetben foglalkoztunk a fejlesztés mögött álló technológiával, valamint az album elkészítésének részletes leírásával. Az ötödik fejezetben pedig összegezzük az eredményeket, és a továbbfejlesztésre teszünk javaslatokat.

Kulcsszavak: digitális képfeldolgozás, képkezelés, háttér téma beállítása, fájlba mentés

Abstract

The goal of the thesis was creating a digital photo album, where the user can upload their images and drag and drop to a book like canvas, custom text can be placed, the images can be rotated, resizable on the page. Users can choose backgrounds, frames, styles, drawings, patterns. After editing it can be saved to pdf file. The thesis consists of 5 main chapters, contains 5 figures and 1 CD annex. In the first and second chapter we describe our goals with the digital album, we also reviewed the alternatives found on the online market. In the third and fourth chapter we showed the technology behind the digital album, also the detailed description of the creation of the album. In the fifth chapter we summarize the results of the development and share ideas on how to improve it.

Keywords: digital image processing, image handling, background adjustment, save to file

Tartalomjegyzék

Bevezetés

A mindennapok során az emberek gyakorta megörökítik kedvenc pillanataikat fényképek formájában. A múltba visszatekintve, amikor még hagyományos fényképezőgépekkel készítettek képeket, meg kellett várni, hogy a film beteljen, időigényes volt a film előhívása, ráadásul a fényképészet teljes egészében költséges is volt. Ráadásul nem volt egyszerűen és olcsón megoldható a saját képekkel kinyomtatott jó minőségű fotóalbum sem, mivel csak nyomdáknak voltak megfelelő felszereléseik a legyártásukhoz. Többnyire a kor trendje az előre legyártott üres fotóalbumok voltak, amikbe behelyezték a meglévő képeiket az emberek. Akkoriban emberek igényének és trendjének még az megfelelt, később ez a digitális korszakban megváltozott.

A digitális fényképezőgépek, kamerás telefonok és az internet széleskörű megjelenése változást hozott. Az egyik nagy előnye az, hogy javíthatunk, korrigálhatunk a digitális képen. A tömörítésnek hála akár kicsi fájlméretet kapunk, de választhatunk a tömörítés szintjén, ami meghatározza a kép minőségét. A digitális eszközök (pl. mobiltelefon, számítógép, laptop) képernyőin ugyan könnyen megtekinthetőek, ráadásként léteznek digitális keretek is, amik megjelenítik a ma készült digitális fotókat, de sokan jobban kedvelik a kézzelfogható minőségi albumokat. Másrészt esztétikusan is mutatnak, és nem utolsó sorban tökéletes ajándék lehet családtagoknak, barátoknak vagy ismerősöknek. Viszonylag olcsón és egyszerűen megvalósítható lett, hogy bárki megszerkeszthetí saját kezűleg saját könyvszerű fotóalbumát egy adott weboldalon, amit megrendelés és elkészülte után megadott címre postáznak a megrendelőnek.

1 Munka célja

Az elkészített munkánk célja egy digitális fotóalbum volt, ami szerkesztés után elmenthető pdf formátumba és végül kinyomtatható bárhol. Ugyan a piacon a munkánkhoz sok hasonló program létezik, de azok általában korlátoltak, mivel az üzemeltetőik nem akarnak olyan funkciókat, ami szerintük kihasználatlan, vagy a felhasználókat összezavarná a használatában. Ellenben a mi munkánkban bizonyos dolgokban nagyobb szabadságra törekedtünk. Például, hogy az elkészített munka elmenthető legyen tetszőleges papírméretben és oldalszámban egyetlen pdf formátumba, akár hozzáadhatunk plusz oldalakat is (tehát, nem előre megadott az oldalak száma). A piacon lévő alkalmazásokban nem kifejezetten engednek minden előbb felsorolt funkciót, abból kifolyólag, hogy csak náluk tudják megvenni az elkészített projektet. Az volt számunkra a legfontosabb, hogy bárhol ki lehessen nyomtatni az elkészített albumot.

A képek feltöltése opció után teljes kontrollra törekedtünk a szerkesztési eszközök hozzáadásánál. Terveztünk méretező, forgatható, filterező, szöveg hozzáadó, és nem utolsósorban fontosnak tartottuk a rajzoló eszközt is. Különböző pecséteket is terveztünk, díszítési lehetőségek céljából. Arra törekedtünk, hogy saját hátteret is lehessen feltölteni. Ezt lehetséges filterezni, eltolni, méretezni, míg a hasonló alkalmazások az eltolást, nagyítást és színváltást tartottak csak fontosnak.

A fejlesztéshez a webböngészőkön futtatható HTML-t választottuk, mivel így széles körben futtatható és platformfüggetlen. A Konva JavaScript könyvtár segítségével oldottuk meg a digitális fotóalbum grafikai felületét, ezek funkcióit, valamint az összes szerkesztési lehetőséget, míg az Uppy JavaScript könyvtár a fájl feltöltéshez volt segítségünkre, a .NET Core backend keretrendszerrel egyetemben.

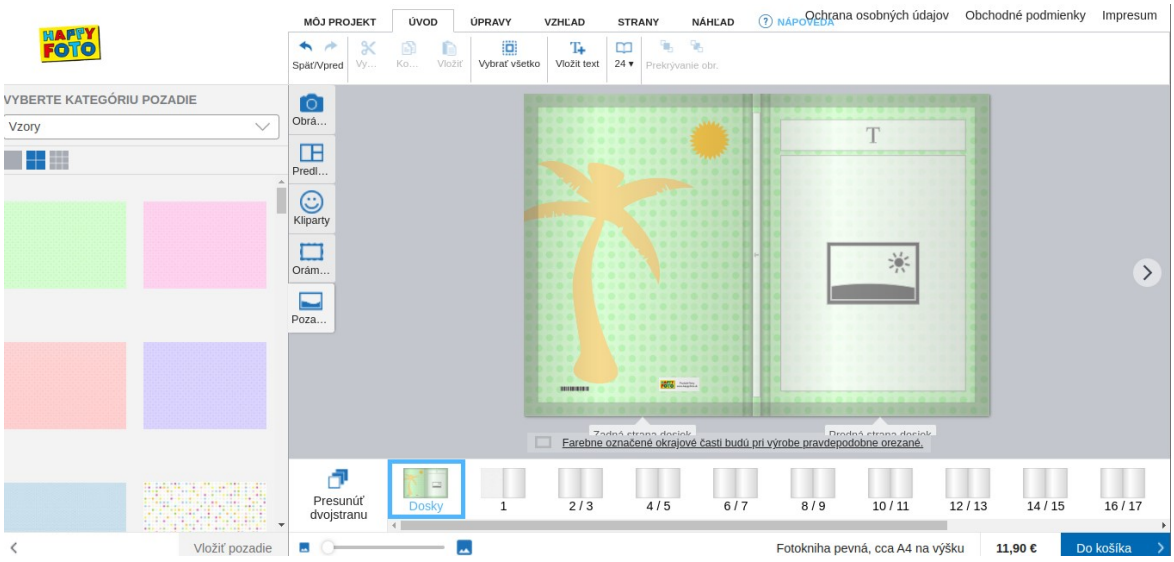
2 Digitális fotóalbumok a piacon

Számos olyan fotószolgáltatás van, ami kínál többek között fotóalbumokat is. A következő két alfejezetben bemutatunk két német vállalatot, ami jelen van a szlovák piacon, és a harmadik alfejezetben egy teljes mértékben szlovák tulajdonú vállalatot. A negyedik alfejezetben pedig összegeztük őket funkcionalitásuk alapján.

A legtöbb fotó szolgáltató az analóg korszak óta jelen van, ami azt bizonyítja, hogy még mindig van igény a munkájukra. Úgy tűnik, hogy egyre több saját fényképes tárgyra van igény, leginkább a versenyképes cégek ki tudják ezt a piaci rést használni. Nem elhanyagolható az sem, hogy a digitális képeket az üzleteikben, vagy automatáikban akár helyben, azonnal kinyomtathatók, ami az analóg korszakban lassú folyamat volt. Lényegében manapság akár saját képeket nyomtatnak bögrére, pólóra, képeslapra, párnára, táskára, és sok más ajándéktárgyra. Bármi elképzelhető és kivitelezhető.

2.1 Happy Foto

A Happy Foto 2004 óta van jelen Szlovákiában, anyacége német tulajdonú. Kínálatuk között különböző fotó szolgáltatások vannak, mint például fénykép előállítás, fotókönyvek, fotófüzet, fotónaptár, fényképes bögrék és egyéb nyomtatott termékek. Általában mindig akad valamilyen promóciójuk, ami által sok új és régi vásárlót késztetnek vásárlásra.



1. ábra: Happy Foto szerkesztő felülete [Forrás: saját kép]

A fotóalbum szerkesztői felülete viszonylag egyszerűen használható, átlátható és nem

utolsó sorban funkciókat tekintve teljes. Szerkeszteni a webes felületen vagy a letöltött számítógépes alkalmazáson (Windows/Mac) illetve, a mobilos alkalmazáson (Android/iOS) lehet. Az album szerkesztő felülete letisztult, átlátható a kezelőfelület. A szerkesztett albumtól közvetlenül balra található 5 fő eszköznek ikonjai, ami az album szerkesztéséhez szolgál, ezek a következők:

- **Képfeltöltés** - Feltölthetők ide a képek, és innen kiválaszthatók.
- **Háttér** - Kiválasztás után áthúzható az albumba.
- **Klipart** - Különböző rajzok választhatók díszítésnek.
- **Keret** - A képek számára különféle keretek választhatóak.
- **Háttérszín** - A hátterek színe módosítható.

Valamelyik opció kiválasztása után, az ikonoktól balra a képernyő szélén egy ablakban jelennek meg a kiválasztott opció lehetőségei, ami bőven nyújt választékot a felhasználó számára az album szerkesztéséhez, vagyis áthúzza bal oldalról a kiválasztott képet vagy mintát, és az album felületére helyezhető.Lehet választani az oldal megjelenő nagyságát az album számára. Ami felhasználói szempontból annyiban előnyös, hogy aki esetleg kicsinek tartja az tudja nagyítani, illetve, aki inkább jobban kedveli, ha egy oldalon több lehetőséget lát, az válassza azt a lehetőséget. A képernyő felső részén találhatók a 7 eszköz nyomógombjai. Egyik kiválasztásánál alatta kis ikonok jelennek meg az adott opcióra. A 7 gomb a következő:

- **Projekt** - A projekt megnyitására és mentésére szolgál, felhőbe vagy lokálisan ment.
- **Kezdőlap** - A kijelölt kép kiválasztható, másolható, kivágható, vagy beilleszthető. Azonkívül a kiválasztott méretben, színben, betűtípusban felirat is itt választható, később rákattintva változtatható.
- **Javítás** - Tartalmaz segédvonalakat, kép törlése, keret módosítása több képre, illetve a képek javítása opciót, ami megjelenít egy külön ablakot. Ez az ablak tartalmaz előre definiált filtereket, de a felhasználó, akár saját maga állíthatja be a fényerőt, kontrasztot, és a gamma korrekciót. A kép képaránya megváltoztatható vágással, vagy előre kiválasztott képaránnyal, vagy kézzel kijelölt képaránnyal, természetesen eltolható a kép a vágás során.
- **Külalak** - Beilleszthető üres képdoboz, az éppen kijelölt doboz pozicionálható a lap oldalának valamelyik széléhez. Az oldalakban hozzáadható plusz két oldal, törölhető

a tartalma, vagy akár az összes oldal megnézhető egyszerre, kis ablakok formájában, egy új lapon.

- **Oldalak** - Hozzáadható plusz két oldal, törölhető a tartalma, vagy akár az összes oldal megnézhető egyszerre, kis ablakok formájában egy új lapon.
- **Előnézet** - Bemutatja, hogy néz ki a kiválasztott szerkesztett kép az albumban, illetve az egész fényképalbum megtekinthető lapozómódban áttekinthetően.
- **Súgó** - Megnyitja a Happy Foto weboldalát tippekért.

Szerkesztő módban az album felületére kattintva lehetséges a képet mozgatni, forgatni, méretezni, nagyítani, tükrözni, kereten belül mozgatni, és nem utolsó sorban törölni a kiválasztott képet. Az album szerkesztői felületének alján vannak még lehetséges az albumot teljes egészében kinagyítani, kiválasztani egy adott lapot, valamint egy új lapon megtekinteni az összes oldalt kis ablak formájában, valamint egy megrendelőgombot is tartalmaz.

2.2 Cewe

A Fotolab partnereként 1995-ben lépett be a szlovák piacra, 2016-ra teljesen kivásárolták őket, és Cewe néven folytatták tevékenységüket. Több terméküket Prágában gyártják le. A német anyacég Európa 24 országában elérhető, a leglefedettebb a kontinensen. Országszerte 8 üzletük van, ahol helyben lehet képeket előhívatni, átvevőhelyek is egyben, illetve fényképészeti segédeszközöket lehet vásárolni. Számos partner üzleteik is vannak országszerte, ahol szintén elő lehet hívatni képeket egy automatából helyben, átvevőhelyként szolgálnak. A legismertebb partnerüzletük a Dm drogéria és a Nay elektronikai üzlet.

Kínálnak fénykép előállítást, fotóalbumot, fotókönyvet, különféle ajándéktárgyakat, mint akár bögréket, pólókat, képeslapokat, párnákat, táskákat, és sok egyéb dolgot. A legnagyobb kínálat az övék ezen a téren.

A szerkesztő kezelőfelülete úgy lett kialakítva, hogy a sok opció ne legyen zavaró az átlagfelhasználó számára. Több opció akkor jelenik meg, ha a felhasználó az albumba behelyezett képre kattint. A webes alkalmazásukon kívül van dedikált számítógépes, mobilos (Android/iOS) alkalmazásuk. Ezek funkcióilag ugyanazt tudják. A mobilos alkalmazás kezelőfelületének dizájnja a kis képernyő miatt leegyszerűsített.

Bal oldalon alapvetően 5 fő eszköz menüpont van, amire kattintva, jobbra megjelenik egy

2. ábra: Cewe szerkesztő felülete [Forrás: saját kép]

ablak a kiválasztható opciókkal, melyek a következők:

- **Fényképek** - A képek feltölthetők és kiválaszthatók.
- **Storybook** - Előre elkészített témák választhatók ki.
- **Szöveg** - Szöveg hozzáadása az albumhoz.
- **Klipart** - Rajzolt díszítőelemek.
- **Hátterek** - A háttérnek szolgáló kép kiválasztható.

A képernyő tetején található opciók nem mások, mint a projekt mentése, az előző opció visszavonása, az album kinagyítása lapozás céljából és pár beállítás. A jobb oldali oldalsáv akkor jelenik meg, ha nincs kiválasztva a bal oldalt található szerkesztői lehetőségekből egy sem. Ha épp elérhető, az oldalakat mutatja kis méretben, átváltható másik lap, illetve plusz lapokat ad az albumhoz, és a megrendelő gombot is elérjük.

Az albumban lévő fotók kattintásra megjelenítenek egy szerkesztő téglalapot, amivel forgatható, méretezhető, törölhető, kivágható az adott kép. Ezenkívül még a bal oldalsávban is megjelennek képszerkesztő eszközök, mint a filterek, keretek, maszkok, stb.

2.3 Albumo

Az Albumo egy viszonylag új szlovák vállalkozás, 2014 óta van jelen. Folyamatos növekedésük hatására Csehországban és Magyarországon is megjelentek. Az előbbi kettő vállalkozáshoz hasonlóan, foglalkoznak fényképek előhívásával, fotókönyvekkel, fotóalbumok-

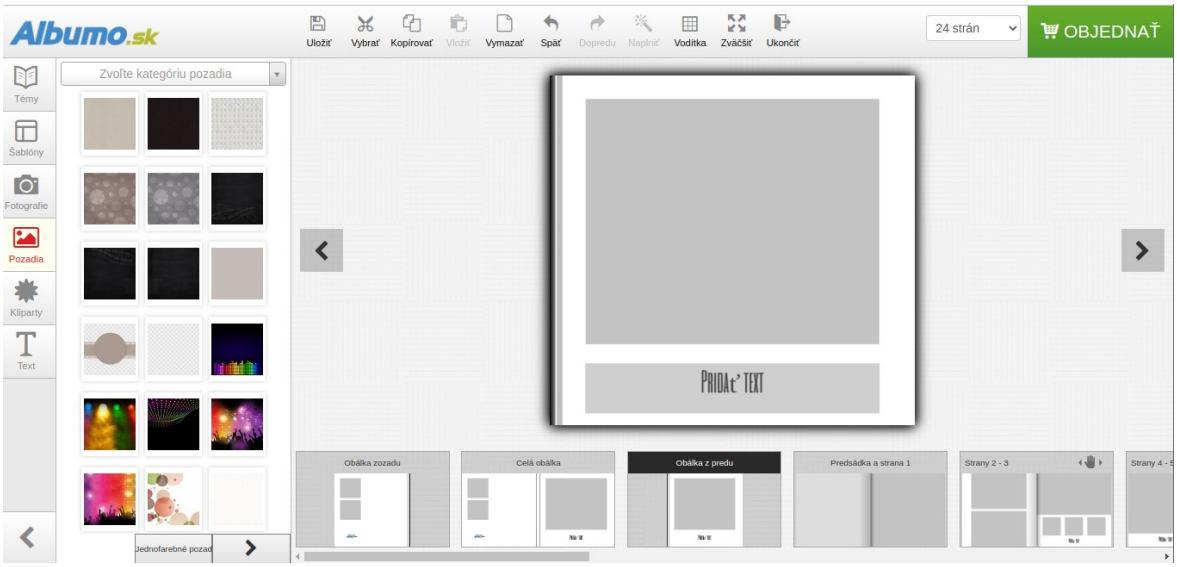
kal, fotóvászonképekkel, valamint nagy formátumú fényképekkel. A kínálatuk kisebb a konkurenciákkal szemben, de a vásárlók így is megtalálják, amit keresnek. Bár ritkán adnak akciót a termékeikre, viszonylag olcsón kínálják termékeiket. Mivel nem rendelkeznek saját üzletekkel, csak weboldalukról rendelhetők a termékeik.

Nincsen saját alkalmazásuk számítógépre, vagy mobiltelefonokra, csupán egy univerzális web alapú böngészős alkalmazás. Ráadásul ez nem kifejezetten mobilbarát, úgyhogy számítógépeken kívül, legfeljebb tableteken használható igazán. A szerkesztő kezelőfelületét próbálták amennyire lehetett automatizálni, az egyszerűbb megvalósítás céljából. Induláskor kész téma opciókat ad, de persze teljesen testre is szabható. Az egyszerűbb technológiai háttér ellenére kapott lapozós animációt is.

A szerkesztőfelület nem kifejezetten a legjobb az oldalsávból az albumba történő behelyezésére. Leginkább csak a képeknél működik egyszerűen, a háttérnél vagy más opcióknál, csak rá kell kattintani és megjelenik. A kezelőfelülete bal oldalon a 6 fő eszköz gombját tartalmazza, az adott opciók közvetlenül a tőle jobbra lévő ablakban jelenik meg. Az opciók a következők:

- **Témák** - Előre elkészített témák kiválaszthatók.
- **Sablonok** - Sablonok kiválaszthatók a képek fotóalbumszerű elrendezéséhez.
- **Fényképek** - A fényképek kiválaszthatók.
- **Hátterek** - A hátterek kiválaszthatók az oldalakhoz, vagy az albumhoz.
- **Klipart** - Díszítőelemekként szolgáló rajzok.
- **Szöveg** - Szöveg illeszthető az albumba, különböző stílusúak.

Tartalmaz még az oldalsáv egy elrejtő gombot, plusz két gombot a képek vagy opciók közötti navigációra. A felület alján kis ablakokban látszódnak az oldalak és azok elrendezéseik, kattintásra az adott oldal megjelenik a szerkesztőben. A felső menüsávban el lehet menteni az albumot az oldal szerverére, amit később vissza lehet állítani. Azonkívül van még kivágó, beillesztő, másoló gomb, ezek a képeken kívül másra is alkalmazhatjuk. Egy törlő gomb teljesen kitörli az addigi munkánkat. A munka visszaállítása miatt kapott egy vissza és egy előre gombot is. Egy automata kitöltő gomb gondoskodik a képek véletlenszerűen való behelyezéséről. Nem utolsósorban a weboldalt kinagyító és bezárás gombokat is tartalmaz, a legvégén egy megrendelő gombot. A szerkesztőfelületen található két gomb az oldalak navigálására is.



3. ábra: Albumo szerkesztő felülete [Forrás: saját kép]

2.4 Összegzés

Elmondható, hogy a három említett és elemzett szolgáltató közül országszerte a Cewe érhető el leginkább, köszönhetően a sok partnerüzleteinek. A kínálata az összesnek nagyjából hasonló, a legnagyobb eltérések közöttük leginkább az áraikban rejlik, bár az eltérő méretek, minőségbeli különbségek, és egyéb eltérő paraméterek és funkciók miatt nem lehet egyöntetűen választani és ítéletet mondani.

A legmegfizethetőbb árú közülük az Albumo, de a Happy Foto gyakran használ akciókat, leárazásokat, ami szintén elgondoltatja a vásárlókat, míg a Cewe közülük átlagban a legdrágább. A legtöbb fajta fotókönyv a Cewe nevéhez fűződik, mivel náluk lehet a legtöbb paraméter szerint választani fotóalbumot, akár prémium keménykötésű lapokat is. A legtöbb terméket is ők kínálják, ebben nem vetekszik senki velük. A Happy Foto sem kínál sokkal kevesebbet, míg az Albumo-nál található a legkevesebb termék, de az átlag vásárló itt is megtalálja, amit keres. Az album szerkesztője és dizájnja szintén a Cewe-nél a legteljesebb, viszont kiemelhető a Happy Foto is, a viszonylag egyszerű, és könnyen használható felületével, az Albumo felülete a legnehezebben átlátható és használóbarát. Attól függetlenül funkcióilag nagyjából ugyanazt kínálják.

Valószínűleg az embereknek leginkább a Cewe neve ugrik be elsőre az országos és európai lefedettségük és kínálatuk miatt. A fotókönyveket is sokféle minőségű és egyéb paraméterekkel legyártott kínálatuk is a legmagasabb. Lényegében az ő marketing tevékenységük a

legnagyobb, de a Happy Foto is elég népszerű, ráadásul ők inkább a webes megrendelőkre építenek. Az Albumo szintén, ők inkább a megfizethető árú termékekre helyezik a hangsúlyt.

A digitális fotóalbumunkban az előbb felsorolt vállalkozásokhoz hasonlóan mi is fontosnak tartottuk a méretező eszközt, filtereket, szövegírást, díszítéseket, kereteket. Viszont azt is fontosnak tartottuk, hogy menthető legyen pdf-be, nyomtatás miatt. Fontos szempont volt az is, hogy legyen rajzoló eszköz is, ha valaki rajzolni szeretne az albumba, hogy ne csak előre megtervezett díszítéseket lehessen alkalmazni. Saját háttereket is lehet a mi digitális fotóalbumunkban választani, ha valaki nem érné be az előre elkészített kínálattal, akár filterezhetőek is.

3 Alkalmazott technológiák a fejlesztés során

A következő alfejezetekben bemutatjuk a digitális fotóalbum elkészítése során felhasznált technológiákat. A HTML és Javascript, illetve nyílt forráskódú Javascript könyvtárakat használtuk. Választhattunk volna más alternatívákat is, de ezek elég hatékonyan és viszonylag elég jól használhatók együtt. A program tartalmaz képfeldolgozási és egyéb hasonló erőforrást nem kímélő folyamatokat, amit a webes technológia frontend-je hajt végre Javascriptben. Ezáltal a szerver oldali backend-nek nem szükséges nagy teljesítményűnek lennie.

3.1 C# és .NET Core 3.1

A C# a Microsoft által fejlesztett negyedik generációs objektumorientált programnyelv. A .NET keretrendszer részeként jött létre, céljuk vele a gyors és hatékony alkalmazás fejlesztés elérése volt. Leginkább a Java és C++ ötvözeteként lehet hivatkozni rá. Azokhoz hasonlóan használ osztályokat, névtereket, interfészeket és tulajdonságokat is. Kiemelhetjük még az úgynevezett „szemétgyűjtő" (garbage collection) funkcióját. Ez felszabadítja azokat az objektumokat a memóriából, amiknek nincs hivatkozásuk.[**5**]

A kép feltöltés miatt elengedhetetlen volt backend-et használni, anélkül nem lehetett volna megvalósítani. A .NET Core-t választottuk egyrészt mivel nyílt forráskódú, másrészt a segítségével akár platform függetlenül (Windows, Linux, macOS) bármilyen webszerverre telepíthető. A .NET Core-ból a web api sablonból hoztuk létre. Szükséges hozzá a C#, C++ vagy az F# nyelv ismerete, amik a népszerűségüknek hála nem nagy akadály, nem kell külön nyelvet megtanulni hozzá, mint a PHP-hoz. Az ebben való fejlesztéshez használható a Visual Studio (19 vagy 22 verziója), akár a sima platformfüggetlen Visual Studio Code is. Szükséges letölteni hozzá a .NET Core csomagot is.

Bár elérhető a .NET 6.0-as verziója is, de ehhez a frissessége miatt még kevés dokumentáció érhető el. Ráadásul évente kap új kiadást is, ami szintén megnehezítette a legújabb verzió választását. A .NET Core 3.1 még szintén támogatott verzió, úgyhogy nem indokolta semmi az újabb verzió használatát. A C# használata is indokolt volt, egyrészt az ajánlott nyelv a .NET-hez, másrészt az egyik leghatékonyabb nyelv az ehhez hasonló backend-ek elkészítésére.

Az Uppy-hoz a C# .NET Core keretrendszerrel készült *Program.cs* kiterjesztésű osztályt

a web alkalmazás mintája automatikusan kigenerálja. Annak a tartalmát nem is szükséges módosítani, mivel annak a fő függvénye a *Startup.cs* kiterjesztés osztályát használtuk a végrehajtandó feladatokhoz. Például a végpontokat is itt kell beállítani, a kérésekre reagáló *Controller*-ekhez. Ez az egyszerű C# példa megmutatja, milyen a frontend felől érkező *onpost* kérés fogadása, ami tulajdonképpen a fájlokat küldi a szerver felé. Ebben az esetben a fájl feltöltést így konfiguráljuk:

```
app.UseEndpoints(endpoints =>
{
endpoints.MapGet("/", context =>
{
context.Response.Redirect("/index.html", permanent: false);
return Task.FromResult(0);
});
endpoints.MapControllers();
endpoints.MapControllerRoute(
name: "Upload",
pattern:"Upload",
defaults: new
{controller="Upload", action= "OnPostUppy" }
);
});
```

3.2 HTML

A neve a Hypertext Markup Language angol nyelvű mozaikszóból ered. Olyan leíró nyelv, amit web oldalak készítésére használunk. Egy HTML kód meghatározza a weboldal felépítését, kinézetét. Webböngészők értelmezik a kódot és megjelenítik a kész oldalakat. Tulajdonképpen leginkább *tag*-ekből áll, amik létrehozzák a strukturált dokumentumot.[**6**]

Mivel a HTML weboldalak önmagukban elég egyszerű kinézetűek, ezért idővel megjelent a CSS stílusleíró nyelv. Ez általánosan a HTML és XHTML leírására jött létre, de akár SVG és XUL leírására is. Megadja a lehetőséget a weboldal stílusának megváltoztatására,

beleértve a háttér színét, betűtípusát, elrendezését, és sok más egyéb stíluselem megváltoztatását. Egy egyszerű példát láthatunk alább, kész HTML oldalról, aminek stílus leírása bele van ágyazva a HTML fájlba**[13]**

```
<html>
<head>
<style>
body {
background-color: lightblue;
}
h1 {
color: white;
text-align: center;
}
p {
font-family: verdana;
font-size: 20px;
}
</style>
<title>Page Title</title>
</head>
<body>
<h1>Ez egy egyszerű fejléc.</h1>
<p>Egy új bekezdés szöveggel.</p>
</body>
</html>
```

A digitális fotóalbumunk alapvetően HTML technológiára épül. Viszont többségében a beleépült JavaScript szkriptnyelvet használtuk a funkciók létrehozására, ezen belül leginkább a *Konva* könyvtárat. Természetesen CSS-t is használtunk, ezeket főleg a gombok és oldalsáv formázásának kialakításához.

3.3 JavaScript

Egy objektumorientált aszinkron szkriptnyelv, amit weboldalakon elterjedten használnak. Különlegessége az egyszerűségében és dinamikus, futási időben történő fordításában rejlik. A böngészők JavaScript motorja hajtja végre a kódot. Ezáltal gyakorlatilag platformfüggetlen nyelv. Manapság a weboldalak elengedhetetlen kelléke, ugyanis nélküle a weboldalak nem lehetnének interaktívak. A HTML eseményekre reagálva végrehajt kiválasztott függvényeket. Léteznek különböző JavaScript keretrendszerek és bővítmények, amik megkönnyítik és hatékonyabbá teszik az elkészített projekteket. [**7**]

Alapvetően a forráskódot beágyazzuk egy *tag*-be a HTML dokumentumba való használatához. Meg lehet adni a *js* kiterjesztésű fájl helyét, vagy egyenesen a HTML *tag*-be is menthető a végrehajtandó kód. Például így:

```
<script src="source.js"></script>
```

Vagy egyszerűen a JavaScript kódot a HTML *script* nevű *tag*-jébe mentjük. Itt egy nagyon egyszerű példa arra, hogyan:

```
<script>
  function HelloWorld(){
  console.log("Hello world!");
  }
</script>
```

3.3.1 Uppy JavaScript könyvtár

Az Uppy egy moduláris Javascript alapú fájl feltöltő könyvtár. A kezelőfelülete elegáns és könnyű használatot biztosít. Nyílt forráskódú, sima böngészőn kívül, modulok által biztosít számos oldalról való feltöltést is: Facebook, Instagam, Dropbox, Google Drive, sok más egyéb. Tulajdonképpen könnyen bővíthető és egy kész felületet biztosít a feltöltéshez, ami kibővíthető egész bonyolult feltöltő felületre is. Szükséges valamilyen bővítményt alkalmazni a böngésző és szerver közötti kapcsolat fenntartásához, mint például az XHR típusú feltöltésnél. [**8**]

A mi munkánk során a XMLHttpRequest(XHR) API segítségét használtuk a feltöltéshez az Uppy-hoz. Ennek az előnye, hogy weblap frissítés nélkül képes kapcsolatot létesíteni

a szerverrel. Lényegében HTTP kérést teljesít. Ezt az API-t leginkább AJAX-ban történő programozásban használják. A HTML fájlban el kell helyezni egy *div*-et ami a feltöltő felületének a helyét jelöli, és erre hivatkoztunk a JavaScript kódban. A nagy előnye az Uppy-nak, hogy a letöltés szüneteltethető a *tus* szabvány által. Fájl kiválasztásánál már a feltöltő gomb lenyomása előtt elkezdi feltölteni a fájl vagy fájlok egy bizonyos részeit az időspórolás végett. A felület és feltöltés inicializálásához ez a kód használható a JavaScript-ben[**9**]:

```
const { XHRUpload } = Uppy;
var uppy = new Uppy.Core()
.use(Uppy.Dashboard, {
   inline: true,
   target: `#drag-drop-area`,
   width: 200,
   height: 200
})
.use(Uppy.XHRUpload, {
   endpoint: `Upload`,
   formData: true,
   fieldName: "files"
})
```

3.3.2 Konva 2d JavaScript canvas könyvtár

A Konva egy nyílt forráskódú objektum orientált 2d-s API, alapvetően HTML5 alapú JavaScript könyvtár. A tökéletes választás lehet vászonra épülő web alkalmazásokra, és képszerkesztésre. Képes objektumként kezelni a képeket. Beépített *transformer* (transzformáló) eszközt tartalmat, amivel az objektum méretei módosíthatók, skálázhatók, forgathatóak. Remekül optimalizált mobilokra is, akár animáció készítésére is alkalmas. Alapértelmezetten tartalmaz filtereket, behelyezhető funkciót, alakzatokat, akár csoportonkénti kijelölést is, saját eseménykezelő rendszere is van. Lehetséges magas felbontású képeket, kép objektumokat exportálni a vásznából. *React* és *Vue* keretrendszert is támogat, ezek segítségével akár egész komoly alkalmazások is készíthetőek. Egyszóval az egyik legjobb választás volt a digitális fotóalbum munkánkhoz, a sok beépített szerkesztési funkciója miatt.

Egy példa a *stage* („színpad”) létrehozásához, ami lényegében az a rész, ahol megjelení-

tődik az elkészített munka:

```
var stage = new Konva.Stage({
   container: 'container',
   width: 1800,
   height: 1800,
});
```

A *stage* objektum *container* tulajdonságába menti el az ugyanolyan nevű *div*-et, ebbe a konténerbe tölti be a *stage*-t, lényegében jelzi a helyét a HTML-ben. A *width*-be és *height*-ba adható meg a mérete (pixelekben).

A következő lépés még egy réteg (layer) hozzáadása, azért fontos, hogy rá lehessen helyezni képeket és objektumokat a későbbiekben. Egy rövid példa erre a következő kód:

```
var layer = new Konva.Layer();
   stage.add(layer);
```

Létrehozása után lényegében hozzáadja a *layer*-t (réteget) a *stage*-be. Teljesítményből kifolyólag legfeljebb 3-5 réteget ajánlanak használni a weboldalon. Bár nem korlátozzák, de annál többnél instabillá válhat. Lényegében egyszerű animációknál lehet a leginkább hasznos. [**10**]

A munkánk során leginkább a Konva-t használtuk, mivel ez teljesítette majdnem az összes megfogalmazott igényünket. Egy remek *canvas* könyvtár, amivel képesek voltunk megalkotni a fotóalbumot. A sima HTML beépített *canvas*-nak elég korlátolt lett volna, a transzformációkhoz külön rendszert kellett volna írni JavaScript-ben. Arról nem is szólva, hogy kiválasztott keretrendszer nélkül nem tudtunk volna képfeldolgozó algoritmikusokat implementálni, míg a Konva ezeket alapból tartalmazta. Az sem hanyagolható el, hogy a weboldalukon számos remekül kidolgozott dokumentáció, segédanyag és demó található vagy elérhető.

3.3.3 jsPDF javascript könyvtár

A jsPDF egy nyílt forráskódú HTML oldalakhoz készült JavaScript könyvtár, ami segítségével pdf-ek készíthetők. Számunkra a mentés és későbbi nyomtatás céljából volt szükség erre a könyvtárra.

Alapvetően JavaScript kóddal lehet létrehozni egy jsPDF objektumot, ami létrehozás után szerkeszthető:

```
var pdf = new jsPDF(slmode, 'pt', [twidth, theight]);
```

Az *slmode* változó tartalmazza a pdf kiválasztott oldaltájolás rövidítését, a második tulajdonságba a *pt* pontokat választunk a nyomtatási nagyság eléréséhez. A *twidth* és *theight* változók tartalmazzák a létrehozott pdf pixel méreteinek a nagyságát.

Az objektum létrehozása után gyakorlatilag szabadon szerkeszthető. A generálás előtt belehelyezhetők képek, szövegek, alakzatok, stb. A munkánk során lementettük először a vászon tartalmát képben, azt helyeztük bele az objektumba, a többi lehetőségre nem volt szükségünk. Pár példa kód a pdf objektum hozzáadásának lehetőségeiről:

```
pdf.setFontSize(fontsize);
pdf.text(x, y, 'Egy jsPdf szöveg');
pdf.addImage(imgData, 'JPEG', 15, 40, 180, 160);
pdf.rect(20, 20, 10, 10);
Az első a szöveg méretét adja meg a f
```

Az első a szöveg méretét adja meg a *fontsize* változóban, míg a második metódus az *x* és *y* változóban megadja a pozícióját. Az idézőjelben meg a kiírandó szöveget. A harmadik hozzáad egy képet a pdf-hez. Az *imgData* változó tartalmaz egy előre beolvasott képet, illetve megadjuk még a kép formátumát, beillesztendő pozícióját és a méreteit. Opcionálisan még akár forgatható paramétert is megengedhetünk. A negyedik egy téglatestet vagy négyzetet rajzol ki, a kiinduló koordinátáival,illetve pixelben megadott méreteivel. De mivel a Konva is tartalmaz alakzat rajzolás lehetőséget, egyenesen lementhető a képernyő tartalma teljes egészében. [**11**]

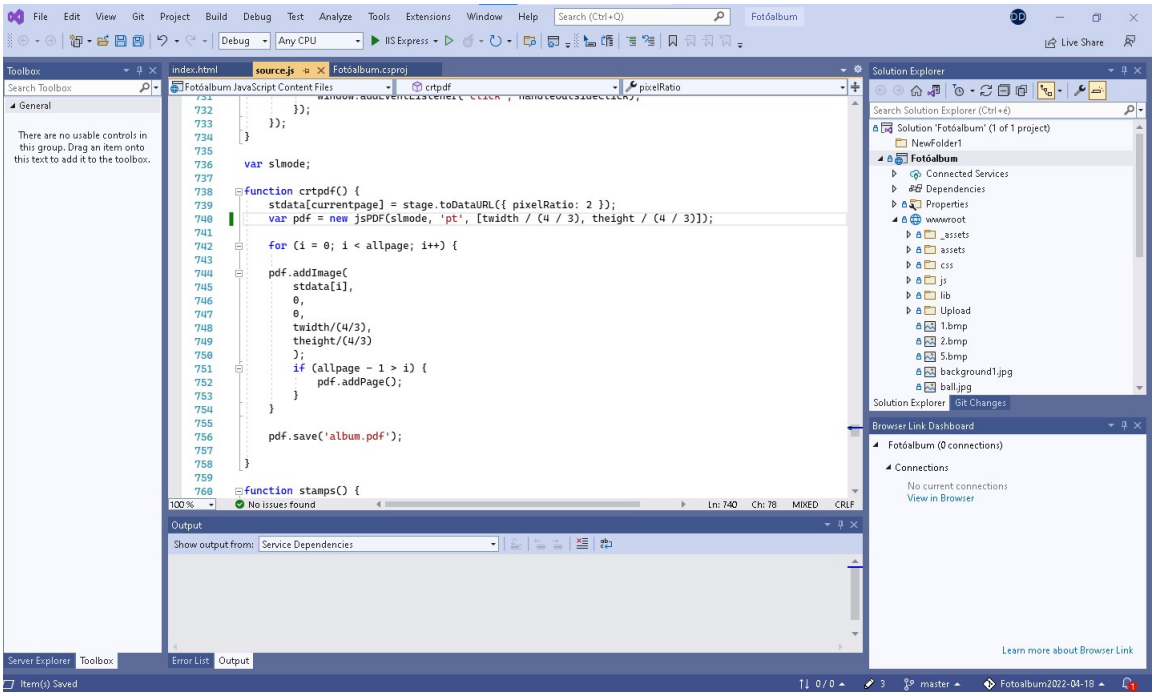
A pdf elmentése előtt nevet lehet adni, a kiterjesztésével együtt, a következőképpen lehetséges:

```
pdf.save('album.pdf');
```

3.4 Visual Studio 2022

A Visual Studio egy több programozási nyelvet tartalmazó fejlesztőkörnyezet (IDE). Leginkább a „freemium” jelző illik rá, ugyanis a Community edition regisztráció után ingyenes,

de a Professional edition és Enterprise edition fizetősek. De az ingyenes változat is szinte ugyanazokat a funkciókat tudja. Annyi megkötés van az ingyenes változatban, hogy oktatásra, vagy kisebb projektekre alkalmazható. A .NET Core keretrendszer is letölthető rá és sok más használt fejlesztési csomag. Köztük többek között C#, C++, UWP, webfejlesztéshez használt csomagok. A nagy előnye fejlesztésnél, hogy ha írunk egy kódot, ad lehetséges opciókat a kódhoz, automatizált. Fejlett hibakereséses rendszere van, monitorozza a program működését, mind az erőforrás használatot. [**12**]



4. ábra: Visual Studio 22 kezelőfelülete [Forrás: saját kép]

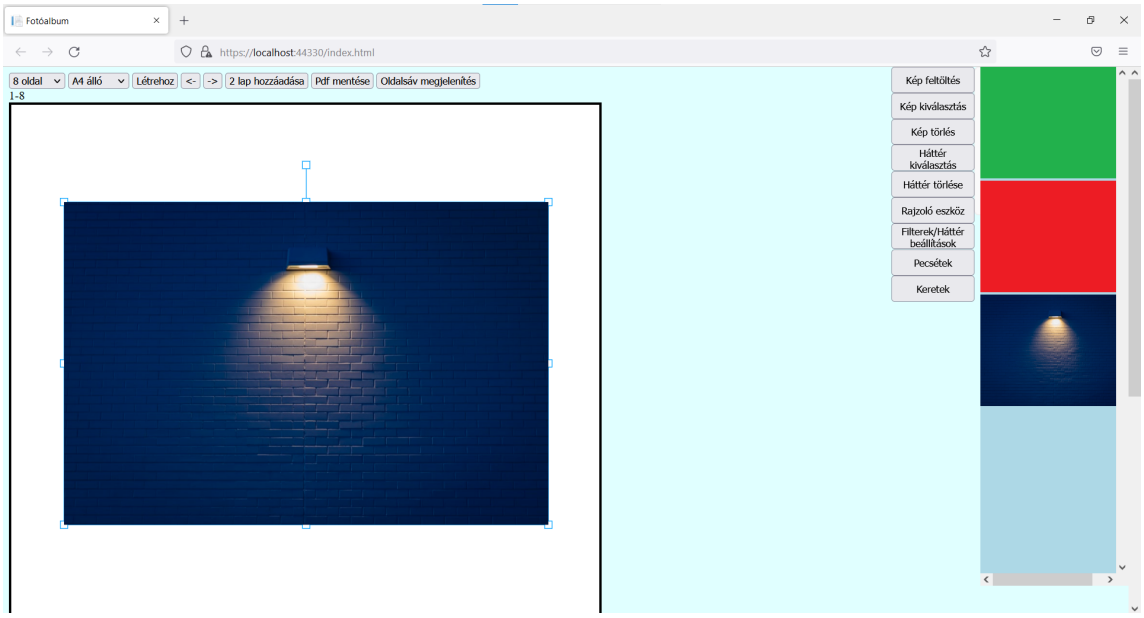
A fejlesztéshez használhattuk volna akár az ingyenes Visual Studio Code-t is. Mivel a Visual Studio sokkal inkább automatizált és megkötések nélkül alkalmazhattuk, ez mellett döntöttünk. Főleg HTML és Konva JavaScript kódot használtunk, amihez többnyire akár egy sima jegyzettömb is elegendő lett volna. Ennek a használata a backend miatt volt szükséges, mivel lehetőség szerint minél kevesebb fejlesztőeszközt szerettünk volna használni.

4 Digitális Fotóalbum kialakítása

A következő alfejezetekben a digitális fotóalbumunk elkészítési menetét írtuk le, bemutattuk részletesen, valamint a működését is részletesebben leírtuk.

4.1 Dizájn

A munkánk során viszonylag egy könnyen átlátható, egyszerű felületre törekedtünk, főleg az album funkcióin volt a hangsúly. Alapértelmezetten a fotóalbum tartalmaz felül egy vízszintes menüsávot. Ez a lista tartalmaz egy legördülő listát, amely megadja az előre definiált oldalszámokat alapértelmezetten. A tőle közvetlenül jobbra lévő lista tartalmaz több előre definiált oldalnagyságot (méretét), a *Létrehoz* nevű gomb ennek megfelelően létrehozza az albumot. Ezeken kívül még két gomb került elhelyezésre a lapozáshoz, ami által lapot vált és a lejjebb lévő lapszám felirat ezek hatására változik. Hozzáadható még két oldal a *2 lap hozzáadása* gomb segítségével. A *Pdf mentése* gomb hatására elmenthetjük az egész fotóalbumot egyetlen többoldalas pdf fájlba. Az utolsó *Oldalsáv megjelenítése* nyomógomb eltünteti, illetve megjeleníti a jobb oldali oldalsávot.



5. ábra: A digitális fotóalbum szerkesztői felülete [Forrás: saját kép]

A jobb oldalsáv tartalmazza az album szerkesztéshez szükséges eszközöket, gyakorlatilag ez a sáv függőleges. Ezen belül is két részre oszlik függőlegesen, az egyik rész a gombokat, a másik a gomblenyomásra megnyílt lehetőségeket tartalmazza. Az első a *Kép feltöltés* ne-

vű gomb megjeleníti a feltöltő felületét, ahová ráhúzhatók vagy kiválaszthatók a feltöltendő képek. Míg a *Kép kiválasztás* megjeleníti az összes feltöltött képet. A *Kép törlés* törli az albumból az éppen kiválasztott képet az album képernyőjén. Hátterek választhatók a meglévő kínálatból, de akár a *Háttér kiválasztása* lenyomásával a feltöltött képekből is áthúzható háttérként. Míg a a *Háttér törlése* opció kitörli az adott oldal háttérét. A *Rajzoló eszközök*-ben található az ecset eszköz rajzoláshoz, radírral együtt. Ugyanitt található a szöveg hozzáadó eszköz, aminél állítható a méret, szín és betűtípus. A *Filterek/Kép beállítások* tartalmaznak különféle képfeldolgozó algoritmusokat, mint a szürkeárnyalat, fényerő és kiemelő csúszkával finomhangolható beállításokat. Ezek innen átállíthatók a háttérre is. Minden más képkezelési beállítások is innen érhetők el, mint például a képek vágása, háttér eltolása vagy nagyságának módosítása. Mivel a rajzolás miatt több objektum elfedi egymást, adtunk opciót az átfedések korrekciójára is ebben a menüpontban. Vannak *Pecsétek* ,amelyek különféle lehetőséget szolgálnak a díszítéshez. Az utolsó *Keretek* nevű gomb néhány egyszerű keretet kínál a képekhez.

4.2 Html elemek létrehozása

A HTML elemeknél nem alkalmaztunk semmiféle sablont vagy weboldali tartalomkészítő rendszert. Ezeket viszonylag egyszerű HTML elemekkel és CSS formázással hoztuk létre. Mivel nem ezeken volt a hangsúly, nem tartottuk fontosnak, hogy keressünk erre is valamilyen különálló keretrendszert. Lényegében a legtöbbet a gombokat, legördülő listát és *div* konténereket használtunk.

A munka során számos *div* konténert használtunk. Ezek külön-külön feleltek lényegében a Konva vászon, Uppy felület, fenti menüsáv és oldalsáv megjelenítéséhez. Ezekből néhányat CSS-ben is formáztunk. Alapvetően így hoztuk létre az oldalsávot, ami ezen belül is tartalmaz *div*-eket, ami a gombok által kiválasztott lehetőségeket ábrázolja:

```
<div id="side" class="side"></div>
```

Az *id* és *class* attribútumok nevüknek megadása után lehet hivatkozni a JavaScript kódban az esetleges változtatások során.

Ezen *div*-en belül találhatóak nyomógombok. A *Képfeltöltés* gombot ily módon hoztuk létre:

```
<button type="button" style="height:35px; width: 112px;"
```

```
onclick="upload()">Kép feltöltés</button>
```

Itt adtuk meg a gomb attribútumait, amint látható itt is megadható a stílus beállítások. Az *onclick* esemény végrehajtja a kijelölt függvényt.

Az oldalsávot úgy állítottuk be, hogy oda rögzítse a képernyő jobb oldalához, a főoldal lapozásától független. Ezt a CSS-el való formázással értük el a következő módon:

```
.side {
    display: flex;
    position: fixed;
    width: 320px;
    height: 700px;
    top: 0;
    right: 0;
}
```

A stílusfájlban a *.side* hivatkozik az azonos nevű HTML elemek osztályára. Ezek méretét a *width* és *height* adta meg, míg a *top* és *right* a felső és alsó margótól számított pozícióját. A *flex* az oldalsáv elemeit széthúzza az oldalsávban, ez nagyban függ az előbb említett méretbeállításokból. A *fixed* az oldalsávot rögzíti az oldalon, lapozás esetén is ugyanott marad.

A filtereknél alkalmaztunk csúszkákat, amit a fényerő beállításánál használtunk. Az *id* megadása után a kódban hivatkozhatunk a beállított értékére. A *range* az értékadó csúszka, megadható a minimuma és maximuma, valamint a lépéstávolságok is. Címkét is alkalmaztunk hozzá, hogy a felhasználó tudja milyen értéket állított át. Ezt így valósítottuk meg:

```
<label for="brighten">Fényerő</label>
<input type="range" min="-1" max="1" step="0.05"
value="0" id="brighten">
```

4.3 Fájlfeltöltés Uppy-val

A fájlfeltöltés megvalósítására azért volt nyilvánvaló választás az Uppy könyvtár, mert hatékony és egyszerűen implementálható volt. Ráadásul továbbfejlesztés esetén sok lehetőséget kínál, például a különböző oldalakról való feltöltést is, vagy több funkciót kínáló felületet.

A backend C# kódjában elsődlegesen beállítottuk a végpontokat, ami a feltöltés felől érkező *OnPost* kérést fogadó *Controller*-hez irányítja át, ahogy egy korábbi fejezetben leírtuk. Az átirányított *UploadController.cs* nevű osztályban dolgozza fel ezt a kérést és egyidejűleg egy megadott mappába menti az Uppy által feltöltött fájlt:

```
public string tmp;
public ActionResult OnPostUppy(List<IFormFile> files)
{
   long size = files.Sum(f => f.Length);
   foreach (var formFile in files)
   {
     if (formFile.Length > 0)
     {
       var filePath = $"wwwroot/Upload/{formFile.FileName}";
       using var stream = System.IO.File.Create(filePath);
       formFile.CopyTo(stream);
       tmp = formFile.FileName;
     }
    }
    return Ok(new {  tmp });
}
```

A *tmp* a fájlnevet tárolja *string* típusú változóként. Ezek visszatérési értékei lesznek a feltöltött fájlnevek *string*-ként és JSON formátumként tárolja el. A böngésző ezeket a JSON fájlneveket válaszként megkapja. A feltöltendő fájlokat a szerver az *Upload* nevű mappába tárolja. Amint látható a fájl létrehozásához a *System.IO.File()* metódust alkalmaztuk, ugyanis a JavaScript fájlok hozzáférése biztonsági okokból korlátolt.

Miután az Uppy sikeresen feltöltötte a fájlt, a JavaScript kódban lefutó *complete* eseménye hozzáadja az éppen feltöltött képeket:

```
uppy.on(`complete`, (result) => {
    console.log("complete", result.succesfull);
    for (let items of str) {
        console.log(items);
```

```
        document.getElementById('sideload').innerHTML +=
         '<img src="Upload\\' + items + '"alt="kep" width="200"
         height="150" onclick="klikk();" draggable="true">';
    }
    str = [];
})
```

Az *innerHTML* beleírta a div konténerbe a kívánt képek HTML attribútumait, ezáltal jelenik meg az oldalsávban. Az *str* nevű tömb tárolja ezen képek neveit. A for ciklussal átnézi az elemeket és az *items* változóba írja az éppen soron lévő nevet, amit az *innerHTML*-be helyezünk. És sorba megjeleníti a feltöltött képeket.

4.4 Konva vászon és képernyő

Alapvetően a Konva JavaScript könyvtár két fő eleme a *stage* és a *layer*, ezek tulajdonképpen egymásra épülnek. A *layer*-ben tároljuk először a betöltött elemeket, ugyanis a rétegek között váltani lehetséges, ami animáció készítéshez is használható. De mi az egyszerűség végett maradtunk az egyetlen réteg használatánál.

Ha az oldalsávból egy képet kiválasztunk és áthúzzuk a képernyőre, célszerű használni a kép címéből való kép objektum létrehozását. Így ugyanis megadható a forráskép elérési helye bármilyen helyről, egy egyszerű JavaScript esemény kéri le az elérési útját kiválasztásnál. Ezeket a következő módon adtuk a réteghez, majd ezután a képernyőre:

```
        Konva.Image.fromURL(itemURL, function (image) {
            image.name('ez');
            image.position({
              x: stage.getPointerPosition().x - (twidth * currentpage),
              y: stage.getPointerPosition().y
            });
            image.src = itemURL;
            image.draggable(true);
            image.cache();
            image.filters([Konva.Filters.Blur,
            Konva.Filters.Brighten, Konva.Filters.Enhance, Grscale]);
```

```
            layer.add(image);
            stage.add(layer);
        });
```

Ez a metódus annyiban különbözik az általános Konva objektumba való betöltéstől, hogy a kép ebben az esetben egy konkrét *Url* címből töltődik be. Ez a módszer a mi esetünkben azért volt hasznos, mert így az oldalsáv képcíméből egyszerűen betöltheti a képet a vászonra, a címe lekérésével. Az *itemURL*-ba jelöli a kép címét, amit egy *dragstart* esemény által kér le a *e.target.src* eseménykezelő objektum. Ezen függvényen belül különböző metódusok adják meg a kép paramétereit. Az *image.draggable(true)* értéket igaz értékre állítjuk, hogy egérrel mozgatható legyen a képernyőn. A kép pozícióját az egérmutató helyzete adja meg. A lapozás miatt az x tengelyen ezt a pozíció pixel értékét kivonjuk a jelenlegi oldalszám és szélességgel. Ezt azért, hogy mindegyik lapra megfelelően adja meg a pozíciót, mivel lapozásnál eltoljuk a megfelelő irányba a rétegen. A filterek metódusába kerülnek a különböző filterek tulajdonságai statikus metódusok által: *[Konva.Filters.Brighten]*. A JavaScript kódban például a *imagesel.brightness(brighten.value)*-val adtuk meg az elmosódás értékét. Az *image.cache()* gyorsítótár használata ebben az esetben ajánlott, ezt tettük mi is. A *layer.add(image)* a rétegre helyezi a képet, míg a *stage.add(layer)* a réteget a vászonra.

A Konva beépített transzformáló eszközt így hoztuk létre egy objektumban:

```
var tr = new Konva.Transformer();
layer.add(tr);
tr.nodes([]);
```

Lényegében hozzáadjuk a réteghez, és a *nodes*-ba adjuk meg a kiválasztandó objektumot, ami alapértelmezetten a mi esetünkben üres volt.

A kijelölő eszközzel egyszerre több kép választható ki és a képeknél megjelenik egyszerre egy közös transzformáló eszköz, ami által egy objektumként méretezhetők, forgathatók, mozgathatók. A kiválasztáshoz használt megjelenő kijelölő téglalapot alapértelmezetten a következőképpen hoztuk létre:

```
var selectionRectangle = new Konva.Rect({
    fill: 'rgba(0,0,255,0.5)',
    visible: false,
});
layer.add(selectionRectangle);
```

Az alapértelmezett szín RGBA kódját megadtuk, alaphelyzetben láthatatlanra állítottuk be, nem utolsó sorban a rétegre helyeztük. A láthatósága és ennek nagysága mindig a kiválasztott képhez vagy képekhez viszonyul, különféle eseményekkel állítottuk át a láthatóságát és pillanatnyi méretét.

4.5 HTML események

HTML elemek attribútumába megadható a végrehajtandó JavaScript kód. Az eseménykezelő reagál a megfelelő esemény végrehajtásával. Akár JavaScript kóddal is hozzárendelhető a kívánt HTML elemhez. A munkánk során ezek használata elengedhetetlen volt, főleg a gombok és a képek áthúzása során.

Az oldalsávból való áthúzásnál sima HTML eseményeket használtunk. Regisztráltuk a képernyőre szánt kép pozícióját a *stage.setPointersPositions(e)* esemény objektummal, lényegében ezzel a vásznon kívüli HTML tartalmaknál manuálisan megadható az egérmutató pozíciója. A kép objektum paramétereit Konva metódusok adták meg. Az elérhetőségi helyét az *itemURL* változóba az *e.target.src* attribútum adta meg *dragstart* esemény által. Ezt a következőképpen oldottuk meg:

```
document
.getElementById('sideload')
.addEventListener('dragstart', function (e) {
itemURL = e.target.src;
});
```

Egy másikfajta eseményfigyelőt adtunk a csúszka *oninput* eseményéhez, a HTML fájl *brighten id*-vel rendelkező div-ét megkereste és külön változóba mentette. Ezzel gyakorlatilag hivatkozhatunk rá, a csúszka értékét a *brighten.value* adja meg:

```
var brighten = document.getElementById('brighten');
brighten.oninput = function () {
imagesel.brightness(brighten.value);
imagesel.setAttr('brighten', brighten.value);
};
```

Mivel a Konva beépített szürkeárnyalatos filtere automatikus több más filterhez képest, kénytelenek voltunk kézzel egy saját algoritmust írni. Aminek alkalmazását a kiválasztott

képhez *if* feltételvizsgálathoz kötöttük. A *Szürkeárnyalat* gombhoz hozzárendeltük a következő *onclick* eseményt, aminek lenyomása után megváltozott a *grayscale boolean* változó értéke:

```
grayscale.onclick = function () {
    if (grscale == true) {
        grscale = false;
    } else {
        grscale = true;
    }
```

A következőképpen adtuk hozzá ezt a *Grayscale* függvényt és minden másfajta Filter beépített függvényét az éppen kiválasztott képhez:

```
imagesel.filters([Konva.Filters.Blur, Konva.Filters.Brighten,
Konva.Filters.Enhance, Grscale]);
```

Ha a *grscale* értéke igaz, akkor a gombnyomásra az esemény végrehajtja a kép szürkeárnyalatossá alakítását a következő algoritmussal:

```
var Grscale = function (imageData) {
    var nPixels = imageData.data.length;
    if (grscale == true)
        for (var i = 0; i < nPixels; i += 4) {
            brightness = 0.34 * imageData.data[i] + 0.5 *
            imageData.data[i + 1] + 0.16 * imageData.data[i + 2];
            imageData.data[i] = brightness;
            imageData.data[i + 1] = brightness;
            imageData.data[i + 2] = brightness;
        }
};
```

Az *imageData* tartalmazza a Konva képobjektum adatait, amit az algoritmus megváltoztat. A *brightness* változó értékét az algoritmus kiszámítja, ami az imageData.data bájtjait megváltoztatja.

4.6 Konva események

A Konva események leginkább csak abban különböznek a HTML eseményektől, hogy annak a vásznához, rétegeihez, illetve objektumaihoz kapcsolódnak az eseményeik. Ilyen esetben is JavaScript függvényt futtat. Sima HTML eseményekkel nem lehet közvetlen módon hozzáférni a vászon tartalmához. Viszont lehetséges regisztrálni manuálisan az egérmutató vásznon kívüli helyzetét is. A *dragstart* HTML esemény lekérte a kép *Url*-ját (címét), hogy a Konva réteghez rendelt eseménye létre tudja hozni a kívánt helyen.

A transzformáló eszközt beépítettük a web alkalmazásba, képre való kattintás után jelenik meg. Ezzel méretezhető, forgatható az albumra lehelyezett kép. Ez az eszköz alapértelmezetten megtalálható a Konva-ban. A következő példában bemutatunk egy részletet, mégpedig a vászonra való kattintás során *click tap* Konva esemény által végrehajtandó kódot:

```
stage.on('click tap', function (e) {
    if (selectionRectangle.visible()) {
        return;
    }
    if (e.target === stage) {
        tr.nodes([]);
        return;
    }
    if (!e.target.hasName('ez')) {
        return;
    }
}
```

A *stage.on()* aktiválja a vászon eseményét, ami akár letiltható is *stage.off()*-al is. Ha a *selectionRectangle.visible()* metódus igaz értéket ad a feltételvizsgálatban, akkor nem csinál semmit a kijelölő téglalap és befejezi a függvény működését. Amennyiben a második feltételvizsgálat igaz, akkor a transzformáló eszköz az összes kijelölését megszünteti és szintén befejezi a függvény működését, visszatérési értéket ad. Ugyanis a *tr.nodes([])* értéke üres, máskülönben a vásznon kiválasztott kép lenne: *tr.nodes([e.target])*. Ha eljut a harmadik feltételvizsgálatig, az leellenőrzi, hogy a kiválasztott alakzatot adja-e meg. Amennyiben az *e.target.hasName('ez')*, neve nem a benne lévő string azonosító, vagyis a feltételvizsgálat

igaz értéket ad, akkor ez is befejezi működését. Ellenkező esetben a mintában már nem látható kód végén a *tr.node([nodes])* már nem üres, az alakzaton vagy képen aktiválva lesz a transzformáló eszköz.

Ugyanilyen típusú transzformáló eszköz alkalmazható szöveg hozzáadásánál is, ugyanúgy ezek is méretezhetők. Viszont ebben az esetben úgy módosítottuk, hogy csak az x tengelyen lehessen méretezhető. Egyrészt, hogy a szöveg képarányát megtartsa, másrészt, hogy feleslegesen ne töltse ki az egész képernyőt. Ezt a következő módon oldottuk meg:

```
var tr = new Konva.Transformer({
    node: textNode,
    enabledAnchors: ['middle-left', 'middle-right'],
    boundBoxFunc: function (oldBox, newBox) {
        newBox.width = Math.max(30, newBox.width);
        return newBox;
    },
});
```

A *textNode* változó maga a beillesztendő szöveg objektuma, amit ezáltal kijelölünk. Az *enabledAnchors* a transzformáló eszköz mozgatható részeit jelöli, itt adtuk meg ezeknek a pozíciójukat a kijelölés során. A *boundBoxFunc* paraméterébe lényegében egy függvény által kiszámoltuk a transzformáló téglalap méreteit úgy, hogy egy picivel nagyobb legyen esztétikai szempontból. Lényegében két számból kiválasztja a nagyobbat a *Math.max()* függvény által.

4.7 Lapok betöltése

Az albumban a lapozás eleinte nagy fejtörést jelentett, ugyanis valami olyan módszert akartunk, ami nem terheli le a memóriát annyira, hogy belassuljon. Eleinte pont ezért egyszerre csak egy oldalt akartunk betölteni. Az album egy oldalát eleinte JSON típusú *string* változóba mentettük el. Ezzel az volt a gond, hogy az oldal betöltésnél több Konva esemény megtört és az esemény visszaállítása után sem működött megfelelően. Ezért inkább azt a módszer választottuk, hogy lapozáskor a *layer*-t (réteget) mindig eltolja a megfelelő irányba.

Új albumnál, a létrehozásánál való réteg betöltésekor kezdeti pozícióban van, az aktuálisan behelyezett elemeket is ilyenkor lényegében betölti az adott oldalú pozícióba. A lapra

behelyezett elemek pozícióját úgy adtuk meg, hogy az adott egérmutató x pozíciójának a helyéből kivontuk az oldal szélessége és adott oldal indexének a szorzatából. Ez a módszer garantálta, hogy a réteg megfelelő helyére helyezze az objektumokat. Ugyanis a vászon pozíciójának mutatója nem terjedt ki a réteg koordinátáira. Egy példa az objektumok pozicionálásához:

```
stage.getPointerPosition().x - (twidth * currentpage)
```

A lapozáshoz használt két nyomógomb függvényei a *next()* és a *previous()*, amelyeket a lapozás, illetve a réteg eltolására használtunk:

```
function next() {
    if (currentpage < allpage - 1) {
        stdata[currentpage] = stage.toDataURL({ pixelRatio: 2 });
        layer.offsetX(layer.offsetX() - twidth);
        currentpage++;
        document.getElementById('felirat').innerHTML =
        currentpage + 1 + "-" + allpage;
    }
}
```

Az *if* feltételvizsgálattal biztosítottuk, hogy ne lépje túl a megengedett számú oldalt és mutassa az aktuális oldalszámot a *felirat* nevű div konténerbe. A *currentpage* pedig az aktuális lap indexe, amit a kép tömbnél használunk. Az *stdata* nevű tömbbe lementi az aktuális oldalt kétszer nagyobb felbontásban képként, amit a szerkesztés végén a pdf-be helyez. A *pixelRatio* felelős a minőségéért, minél nagyobb érték, annál jobb minőségben menti le, de a teljesítmény rovására. A réteget eltolja úgy, hogy kivonja az éppen aktuális x tengelyen lévő pozíciójából a vászon szélességét. Visszafelé lapozáskor pedig hozzáadja a szélességét. És visszafelé való lapozáskor ilyen feltétellel biztosítottuk a számozás betartására:

```
if (currentpage > 0)
```

4.7.1 Lapok létrehozása

A lapok számát a *select* HTML elemmel adjuk meg alapértelmezetten, a *page.value* attribútuma tárolja a számát. Amit az album létrehozó függvénye a *Létrehozás* gomb lenyomására, a következő JavaScript kód globális változójának adja meg:

```
allpage = parseInt(page.value);
```

Így gyakorlatilag elmenti a lapok számát egy *allpage* nevű változóba, amihez igény esetén hozzáadhatunk két lapot. Ugyanezt a változót felhasználjuk az oldalak lementésénél használt tömb maximális index méreténél, ezek reprezentálják az oldalak számát. A *currentpage* nevű változóba tároljuk az aktuális oldalszámot, ami új album létrehozásánál felveszi a nulla értéket, és lapozásnál egyel növeli vagy csökkenti, ezt a változót később a pdf mentésnél használjuk. Ugyanis az adott képeket tartalmazó tömb indexe, és így rámutat, hogy melyik oldalt írjuk felül.

Az papírméretek és tájolás megadása után a *createalbum()* függvény létrehozza az ahhoz viszonyított oldalt a megfelelő méretarányos pixelnagyságban. Ezek szélességét, magasságát és tájolását megadtuk, egy *if, else if* feltétellel. A feltételek a HTML legördülő listájában lévő opció szám *id*-jéhez kötődnek és a *size.value* adja meg. Ez a lista általunk előre megadott papírméreteket tartalmaz. Mivel a pontosság kedvéért *float* típusú változót kellet volna megadni, viszont az ezzel való osztás pixelméretben való átváltásnál költséges erőforrású. Arról nem is beszélve, hogy felesleges volt túl kicsi vagy túl nagy papírméretekkel foglalkoznunk, ezért adtunk meg több nyomtatás során kivitelezhető méretet. A feltétel a következőképpen válassza ki az adott opciót:

```
if (size.value == 1) {
    twidth = 793.70;
    theight = 1122.52;
    slmode = 'p';
}
```

4.8 PDF-be való elmentés

Egy korábbi fejezetben bemutattuk a jsPDF pdf generáló könyvtárat. Ebben a fejezetben részleteztük, hogyan tudtuk ezt a Konva-ban készült album elmentésére használni. Először is be kellet tölteni a HTML dokumentum *<script> tag*-jébe a kód forrásának a helyét. Alapvetően ezt a JavaScript függvényt használtuk ennek az elérésére:

```
function crtpdf() {
    stdata[currentpage] = stage.toDataURL({ pixelRatio: 2 });
    var pdf = new jsPDF(slmode, 'pt', [twidth / (4 / 3),
```

```
theight / (4 / 3)]);
for (i = 0; i < allpage; i++) {
pdf.addImage(
    stdata[i],
    0,
    0,
    twidth/(4/3),
    theight/(4/3)
    );
    if (allpage - 1 > i) {
        pdf.addPage();
    }
}
pdf.save('album.pdf');
}
```

Létrehoztunk ebben egy *pdf* nevű objektumot, lényegében ezt használtuk a fájl létrehozásához. Az *stdata* tömbbe elmentettük az adott oldal vásznát kép alapú *Data Url*-ba, a többi oldalt a lapozások *next()* és *previous()* függvények mentették el. Alapértelmezetten az albumot létrehozó függvény elmenti a vászon kezdeti üres állapotát a megadott oldalnyi képbe, amit az előbb említett függvények változáskor felülírnak új képpel róluk, így biztosított, hogy a megadott oldalnyi lapszámú legyen a pdf dokumentum, mindig frissíti, ha szükséges. A pdf létrehozása előtt azért használtuk, hogy el lehessen menteni az aktuálisan kiválasztott oldal változásait is. A *pixelRatio* tulajdonsága jelzi az elmentett kép felbontás minőségét, ez esetben kétszer nagyobb felbontásban menti el a vászonhoz képest.

Mivel a jsPDF-ben való elmentés során a pixelben megadott érték nem a kívánt papírméretet adta meg, ezért az album pixel értékeit átkonvertáltuk pontokra. Lényegében a *twidth* és *theight* a vászon méreteit reprezentáló változók ezért osztottuk el négyharmaddal, az átváltási érték körülbelül 1.3333 érték. Az *slmode* a kiválasztott oldaltájolási módot jelöli. Egy for ciklus segítségével és a *pdf.addImage()* metódus által hozzáadtuk a vászonról készült képeket, amelyeket egy *stdata* nevű tömb tárol, ami minden iterációnál más oldalt helyez a pdf-be. A for ciklus határait az *allpage* nevű változó jelenti, aminek az értékét a HTML fájlból nyertük ki, megváltozik lap hozzáadásánál. A ciklus végén mindig egy új lapot adtunk hozzá a

pdf.addPage() nevű metódussal, amit egy if elágazás vezérel, a második iteráció után fut le, hogy ne legyen feleslegesen üres lap.

5 Fejlesztés során szerzett tapasztalatok és eredmények

Miután a kívánt funkciókat sikerült működőképessé tennünk és a tesztelés során felmerülő hibákat sikeresen kijavítottuk, a programot ezek után elkészültnek nyilvánítottuk. Mivel webalkalmazás, a terjesztése interneten könnyen megvalósítható weboldal formájában valamilyen webszerverről.

5.1 Felmerülő problémák a fejlesztés során

Mint bármilyen más alkalmazás fejlesztésnél, nálunk is jelentek meg problémák, gyakran ütköztünk korlátokba. Ezért előfordult, hogy változtatni kellet, vagy más módszer használata volt szükségszerű.

Általánosan a leggyakrabban előjövő hibákat a Konva eseményei okozták. Ugyanis nem mindig működött együtt több ilyen esemény. Próbálkoztunk azzal, hogy egy adott folyamatnál csak az ahhoz tartozó események fussanak és a többit ideiglenesen kikapcsoltuk. Csakhogy az újraaktivált esemény nem tárolta el a deaktiválása előtti állapotot, és csak az újonnan szerkesztett elemeken működtek. Ezen problémák többségét feltételekkel kezdtük szabályozni, amik többségében sikeresnek bizonyultak.

Az album pdf-be való mentése is okozott némi fejtörést, ugyanis nehézséget okozott, hogy mentsük el a lapok pillanatnyi állapotát. Ugyanis, egyszerre nehezen lett volna kivitelezhető, hogy az összes oldalt egyszerre mentse el, az eléggé lassú lett volna. Arról nem is beszélve, hogy nem lett volna esztétikus sem, ha sorra tölti be pdf mentéskor a képernyőre az összes oldalt. Ezért azt a megoldást találtuk, hogy elmentsük minden lapozásnál és a pdf elmentésénél is az éppen képernyőn lévő oldalt. Bár több oldalnál így is néha kicsit lassan menti el a végén, amikor több nagy felbontású képet kell elmentenie, de ezekért sokszor a JavaScript nyelv korlátai tehetnek.

5.2 Továbbfejlesztési javaslatok

Mivel a Konva elég sokszínű JavaScript könyvtár, elég sok kiaknázatlan lehetőség van. Elsődlegesen az előre elkészített hátterek, pecsétek, illetve rajzeszközök lehetőségeit lehetne bővíteni, ugyanis a piacon lévő versenytársakhoz képest nem olyan gazdag. Mivel alkalmazhatók képfeldolgozó algoritmusok, a filterek lehetőségeit is lehetne bővíteni különböző

hasznos és a Konva-ban alapértelmezetten nem található filterekkel is.

Lehetséges cél megvalósítani a JSON fájlba való mentést, amivel lényegében elmenthető a projekt, majd visszaállítható későbbi szerkesztéshez. A fejlesztés során teszteltünk ehhez hasonló törekvéseket a lapok betöltéséhez. Viszont az események nem úgy működtek, ahogy számítottunk rá, ráadásul az időhiány is beleszólt ezen törekvéseinkbe. Attól függetlenül, ha megfelelően hozzá lenne igazítva az eseményekhez és Konva réteghez a JSON fájlból való betöltés, akkor megvalósítható.

A felhasználói felületen is lehetne csiszolni, mivel viszonylag egyszerű HTML elemeket alkalmaztunk. Esetlegesen valamiféle szerkesztő keretrendszer is alkalmazható lenne. Esetlegesen a gombok és elemek interaktívabb megjelenésére is lehetne törekedni. A szöveg beillesztésénél például meg lehetne oldani, hogy nem csak az oldalsávból lehessen változtatni a paramétereit.

Nem utolsó sorban az oldalt hozzá lehetne igazítani még inkább mobil eszközökhöz is. Ezek számára biztosítani lehetne egy kényelmes, egyszerűen használható felületet kis képernyőn. Egy olyan funkciót, ahol fényképezni lehet és egyenesen feltölti a képet a szerkeszteni kívánt albumba.

Befejezés

Az elkészült webes alkalmazás lehetőséget nyújt a felhasználónak saját maga megtervezett fotóalbum megtervezésére és nyomtatásra használható pdf-be való mentésre. Általánosan a kitűzött céljainkat sikerült elérni, habár volt olyan funkció, mint például a keretek, amelyeknél viszonylag leegyszerűsített megoldást alkalmaztunk. Viszont sikeresen megoldottuk, hogy a képeket a transzformációs eszközzel igény szerint méretezhetjük, forgathatjuk és akár finomhangolhatjuk filterekkel.

A továbbfejlesztési lehetőségek megvalósítása után egy igazán fejlett webes alkalmazás lehet a fotóalbumunk. Amennyiben a projekt elmentése megoldott lesz és alapértelmezetten több előre elkészített témát, valamint pecsétet tartalmaz, akkor válhat igazán kiemelkedővé.

A felhasznált technológiák által a szerző megismerkedett a web alkalmazások és képszerkesztésből adódó lehetőségekkel. Az ezáltal nyert tudással a további célok is megvalósíthatók, további időráfordítással.

Resume

Ľudia si v dnešnej dobe veľmi radi zanechávajú svoje najkrajšie spomienky v podobe fotiek. V minulosti to bol zdĺhavý proces, bolo treba čakať na to, kým sa film naplní a vyvolanie fotiek tiež nebolo rýchle. Nehovoriac o tom, že fotografovanie bolo veľmi drahou záležitosťou. V tomto čase ani fotoalbum s vlastnými fotkami nebolo samozrejmosťou. Ľudia vtedy mohli mať doma iba vopred zhotovené albumy, do ktorých bolo možné vložiť také fotky, ktoré vlastnili doma. Keď zavítalo obdobie digitálnej fotografie, požiadavky ľudí sa zmenili a vlastné digitálne fotoalbumy sa stali veľmi obľúbeným.

Práca obsahuje päť kapitol a mnohé podkapitoly a je ukončený záverom a hodnotením.

V prvej kapitole sme opísali naše ciele to, že v akom formáte sme predstavovali náš fotoalbum. Cieľom tejto práce bolo vytvoriť taký digitálny fotoalbum, kde je možné umiestniť nahrané fotky na knižnej ploche, a na fotkách je možné umiesntiť rôzne nápisy, je tiež možné ich polohovať a nastaviť ich veľkosť. Je možné zvoliť aj pozadie, rám, rôzne štýly, tvary, šablóny. Nakoniec je možné aj uložiť do formátu pdf. Existujú tomuto podobné programy, ale tvorcovia týchto programov považujú tieto funkcie za bezvýznamné. My v našej práci sme sa snažili o väčšiu voľnosť. Hotovú prácu je možné uložiť v pdf formáte pri akejkoľvek veľkosti papiera a rozsahu, ale je možné pridať aj strany navyše. Na trhu dostupných aplikáciách nie je možné využiť všetky spomenuté funkcie z toho dôvodu, aby sme kúpili u výrobcu už celkový hotový projekt. Pre nás bolo najdôležitejšie, aby sme album vedeli kdekoľvek vytlačiť.

V druhej kapitole sme zrealizovali prieskum na slovenskom trhu, boli sme zvedaví aké fotoslužby existujú s dôrazom na výrobcov. V Európe najširšie prekrytie má Happy foto a Cewe. Našli sme aj firmu Albumo. Albumo je čerstvá slovenská firma, ktorá zacielila na strednú a východnú Európu. Nakoniec sme porovnali ich prácu a poskytované služby. Popísali sme aj to, čo považujeme za ~~naše~~ prednosti oproti týmto službám. Konkurentné programy nasledujú zásadu, aby zabezpečili automatizované prostredie bez zbytočných funkcií. Je možné skonštatovať, že každá ponuka je veľmi podobná, s minimálnym rozdielom pri edičnej ploche. Tieto firmy ponúkajú aj rôzne vlastné obrázkové produkty, práve preto sa nesústreďujú na zlepšenie edičných plôch. Pre nás okrem uloženia do pdf formátu, boli dôležité aj nástroje na kreslenie a aby boli voliteľné nie iba predlohy. V našom fotoalbume

je možné zvoliť aj vlastné pozadie v prípade, že niekto potrebujesi to upraviť aj pomocou filtrov.

V tretej kapitole sme prezentovali použité technológie, opisy a kódy o ich použití.

K rozvíjaniu sme vybrali HTML, pretože sme brali ohľad na praktickosť a nezávislosť tejto webovej aplikácie. Grafickú plochu digitálneho fotoalbumu sme vybavili pomocou Konva JavaScript knižnice a tiež všetky jej funkcie a možnosti editácie. Pri nahratí súboru nám pomáhala Uppy JavaScript knižnica, s .NET Core backend systémom. Celkový rozvoj sme vykonávali v programe Visual Studio 22 kvôli backendu a kvôli tomu, aby celá práca mohla byť hotová v jednotnom prostredí.

V procese rozvoja najpoužívaňejšou knižnicou bola Konva v rámci JavaScriptu, pretože pri spracovaní fotografie obsahuje aj zabudované filtre. Ku takzvaným plátnam sme používali vrstvy, pretože pri listovaní nám posunú vrstvy do požadovaného smeru. Pomocou jsPDF knižnice sme mohli album uložiť podľa adekvátnych parametrov do pdf formátu.

V štvrtej kapitole sme sa zaoberali s našou prácou v širšom zmysle. Popísali sme ako sme vedeli vyhotoviť html prvky fotoalbumu, JavaScript kódy, CSS prvky a backend u .NET CORE. Najprv sme predstavili ~~jeho~~ design a pomocou obrázku sme opísali jeho vlastnosti a technické vyhotovenie od Uppy nahratia až ku objaveniu sa na obrazovke. Okrem toho sme opísali aj rôzne kódy pri HTML a Konva a tiež možnosť načítania strán a listovania. Umožnili sme aj uloženie projektu do formátu PDF. Problémy nám spôsobili bariéry JavaScript programu a tiež nespratnosti Konva akcie.

Počas našej práce sme sa snažili vytvoriť jednoduchú a prehľadnú plochu s dôrazom na funkcie albumu. Fotoalbum v základnej doméne obsahuje vodorovné menu. Tento zoznam obsahuje ešte ďalší zoznam, ktorý nám určuje preddefinované čísla strán. Zoznam, ktorý sa nachádza na pravej strane, obsahuje viac preddefinovaných veľkostí strán. Tlačidlo *Vytvoriť* podľa možností vytvorí album. Okrem týchto sa tu nachádzajú ďalšie dve tlačidlá, ktoré zmenia stranu a podľa toho sa zmení aj nápis. Je možné pridať ešte dalšie strany pomocou tlačidla *Pridať dve strany*. Na pokyn tlačidla *Pdf* sa celý fotoalbum uloží do viacstranového pdf súboru. Posledné tlačidlo skryje alebo zobrazí pravý bočný panel.

Pravý bočný panel obsahuje nástroje na editovanie. Tento panel je zvislý. Podľa toho sa zvislo delí na dve časti, jedna časť obsahuje tlačidlá a druhá zobrazuje ich možnosti. Prvé

tlačidlo s názvom *Nahratie obrázku* znázorňuje plochu nahratia, na ktorú je možné uložiť ťahaním zvolené fotky. Tlačidlo *Výber fotky* znázorňuje všetky nahraté fotky. *Vymazanie fotky* odstráni zvolenú fotku na ploche albumu. Pozadie je možné vybrať z už existujúcich pozadí, alebo tlačidlo *Vybrať pozadie* nám dáva možnosť zhotoviť pozadie z nahratých fotiek. Výber tlačidla *Vymazanie pozadia* odstráni pozadie danej strany. Nástroje kreslenia a guma sa nachádzajú pri možnosti *Nástroje kreslenia*. Nájdeme tu aj ponuku pridania textu, pri ktorom je možné nastaviť veľkosť, farbu a typ písma. Nastavenie *Filtra/Obrázky* obsahujú rôzne algoritmy, ako pomer sivej farby, tón svetla a ďalšie možnosti nastavenia. Tieto platia aj pri pozadiach. Ostatné nastavenia fotiek sú tiež dostupné ako napríklad orezanie obrázkov, posun pozadia alebo zmena veľkosti. Kvôli tomu, že pri kreslení viac objektov sa môže prekrývať, pridali sme opciu na korekciu tohto javu. Existujú aj *Pečiatky* s rôznymi možnosťami dekorácie. Posledné tlačidlo s názvom *Rámy* poskytuje niekoľko jednoduchých rámov k obrázkam.

V piatej kapitole sme prezentovali dosiahnuté výsledky a nadobodnuté skúsenosti pri tomto projekte. Navrhovali sme nápady na rozvíjanie programu ako napríklad optimalizovanie na mobilné zariadenia. Algoritmy by bolo možné doplniť aj s vlastnými filtrami. Je dôležitá aj možnosť uloženia do JSON formátu s cieľom ďalšieho editovania. Počas rozvíjania sme narazili na problém nefungujúcich Konva akcií alebo uloženia strán albumov do PDF formátu.

Záver obsahuje všetko, čo sa nám podarilo zrealizovať z výtýčených cieľov.

Ďalej sme zdôraznili hotové opcie editovania. Táto webová aplikácia ponúka používateľom možnosť vytvorenia vlastného digitálneho fotoalbumu, taktiež vytlačenia a uloženia v pdf formáte. Naše vytýčené ciele sa nám podarili dosiahnuť, avšak pri funkcii Rámy sme aplikovali jednoduchšie riešenie. Napriek tomu sme úspešne vyriešili, aby bolo možné pri obrázkoch s nástrojmi nastaviť ich veľkosť, polohovanie a ladenie filtrov. S ďalšími možnostami rozvoja môže byť náš fotoalbum naozaj skvelý aj ako apikácia. V prípade, že projekt bude obsahovať viac základných preddefinovaných tém a tiež bude obsahovať pečiatky, tak v tomto prípade sa môže stať naozaj významným.

Irodalomjegyzék

Online források:

[1] https://blogs.perficient.com/2015/04/29/jspdf-in-web-portal/ [letöltve: 2022.04.27]

[2] https://konvajs.org/ [letöltve: 2022.04.28]

[3] https://www.cewe.sk/sposob-objednania [letöltve: 2022.04.28]

[4] https://www.albumo.sk/fotoknihy.html [letöltve: 2022.04.28]

[5] https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/ [letöltve: 2022.04.14]

[6] https://developer.mozilla.org/en-US/docs/Web/HTML [letöltve: 2022.05.02]

[7] https://developer.mozilla.org/en-US/docs/Web/JavaScript [letöltve: 2022.05.02]

[8] https://uppy.io/ [letöltve: 2022.04.20]

[9] https://uppy.io/docs/tus/ [letöltve: 2022.04.20]

[10] https://konvajs.org/docs/performance/Layer_Management.html [letöltve: 2022.04.16]

[11] https://parall.ax/products/jspdf [letöltve: 2022.04.27]

[12] https://visualstudio.microsoft.com/vs/ [letöltve: 2022.04.30]

Könyv alapú források:

[13] DUCKETT, J. HTML & CSS : Desing and Build Websites: John Wiley & Sons. ISBN 978-1-118-00818-8

Melléklet

A bekötött munka példányának hátsó belső borítójához rögzített CD tartalma:

Demeter-Dávid.pdf

Digitálisfotóalbum.rar (munka gyakorlati része)