Tartalomjegyzék

[1 Bevezetés 1](#_Toc149826946)

[2 Hasonló rendszerek vizsgálata 2](#_Toc149826947)

[2.1 Felhőalapú fájlszolgáltatások 2](#_Toc149826948)

[2.2 Pixieset 3](#_Toc149826949)

[2.3 Iskolaévkönyv 4](#_Toc149826950)

[2.4 Összegzés 5](#_Toc149826951)

[3 Technológiák vizsgálata 6](#_Toc149826952)

[3.1 Backend környezetek 6](#_Toc149826953)

[3.1.1 PHP 6](#_Toc149826954)

[3.1.2 Python 7](#_Toc149826955)

[3.1.3 .NET 8](#_Toc149826956)

[3.1.4 Összefoglalás 10](#_Toc149826957)

[3.2 Frontend környezetek 10](#_Toc149826958)

[3.2.1 Angular 11](#_Toc149826959)

[3.2.2 Vue.js 11](#_Toc149826960)

[3.2.3 React 12](#_Toc149826961)

[3.2.4 Összefoglalás 13](#_Toc149826962)

[4 Használt technológiák ismertetése 14](#_Toc149826963)

[4.1 Backend technológiák 14](#_Toc149826964)

[4.1.1 SimplePay 14](#_Toc149826965)

[4.2 Frontend technológiák 14](#_Toc149826966)

[4.2.1 TypeScript 14](#_Toc149826967)

[4.2.2 ESLint 14](#_Toc149826968)

[4.2.3 React 14](#_Toc149826969)

[4.2.4 Redux 17](#_Toc149826970)

[4.2.5 Axios 17](#_Toc149826971)

[4.2.6 MaterialUI 17](#_Toc149826972)

[5 Irodalomjegyzék 19](#_Toc149826973)

# Bevezetés

Még emlékszem, ahogy általános- és középiskolás éveim során nagy lázban égett az egész iskola – hiszen jöttek az iskolai fotósok. Az elkészült képeket hetekkel később miniatűr, A5-ös méretű lapra nyomtatva küldték el, majd ezeket a lapokat otthon a gyerekek a szüleik kezébe nyomták. A szülők nagyítókkal kiválogatták a jól sikerült képeket, majd a rendelésüket egy megadott határidő előtt visszaküldték az iskolába a megfelelő készpénz mennyiség mellett. Nem volt egy egyszerű folyamat, gyakori volt az emberi hibából adódó hiba. Személyesen tapasztaltam, hogy hibás képek érkeztek a fényképésztől, mások túl kevés példányt kaptak, vagy éppen rossz pénzösszeget küldtek be a szülők.

Az iskolai éveim már régen véget értek, az iskolai fotózás csak egy halvány emlékként maradt fent. Azonban családi vonatkozásban újra előkerült a téma, ahol egy iskolafotózással foglalkozó családtagom hosszasan taglalta, milyen idejétmúlt már a fent leírt rendszer – melyet ő, és sok munkatársa a mai napig használnak. Ekkor fogalmazódott meg bennem, hogy egy olyan platformot szeretnék kiépíteni, ami könnyebbé teszi a fényképész munkáját, valamint a szülőknek is egy felhasználóbarát környezetet biztosít a fényképek megtekintéséhez és megrendeléséhez. Volt, hogy már hozzá is láttam ennek a felületnek az elkészítéséhez, de az akkori tudásom nem volt elegendő egy igazán flexibilis rendszer megtervezéséhez.

A weboldal bonyolultsága több okból is ered. Egy olyan rendszert kell kiépíteni, ahol a fényképész képes külön kezelni az egy adott tanulóról készült képeket, valamint azokat megosztani a szülőkkel. A fotósnak külön intézményeket kell tudni kezelnie, képeket feltölteni a rendszerbe, a rendelések adatait és állapotát könnyen kezelni, és megadni, hogy milyen méretű képek vagy típusú ajándéktárgyak rendelhetők jelenleg. A szülők számára egy letisztult, felhasználóbarát vásárlási felületet kell biztosítani. A weboldalra érkezve meg kell adniuk a fotós által hozzájuk eljuttatott egyedi kódot. Ezt beírva a weboldal kilistázza azokat a képeket, melyen az ő gyermekük látható. Más fényképet nem láthatnak ezen az oldalon.

A rendszer kiépítéséhez backenden ASP.NET Core keretrendszert, a frontend oldalon pedig a React-et választottam.

# Hasonló rendszerek vizsgálata

Ebben a fejezetben össze szeretném hasonlítani azokat az informatikai rendszereket, melyeket a fotósok jelenleg is használhatnak arra a célra, hogy az elkészített fényképeket megosszák a szülőkkel egy iskolai fotózás után. Mindegyik rendszernél megvizsgálom, hogy mennyire alkalmas a képek megosztására, valamint ad-e lehetőséget a képek megrendelésére.

## Felhőalapú fájlszolgáltatások

Valójában a mindenki által ismert felhőalapú fájlszolgáltatások (mint például a OneDrive, Dropbox, Google Drive, stb.) is alkalmasak lennének a fényképek megosztására a szülőkkel. Ezek a platformok lehetővé teszik, hogy tetszőleges mennyiségű adatot osszunk meg bárkivel, és mindeközben az adatvesztés kockázata közel nulla. De ahhoz, hogy megvizsgáljuk, hogy mennyire tudjuk kiváltani a fényképmegosztó portálunk funkcióit hasonló megoldásokkal, meg kel vizsgálnunk milyen hátrányai és előnyei vannak ezeknek a rendszereknek.

A fenti weboldalak mindegyikén a fényképész fel tudja tölteni a fényképeket (vagy azok vízjellel ellátott miniatűrjét), majd azokat gyerekenként mappákra tudja osztani. Ez után az adott gyermekhez tartozó mappát meg tudja osztani csak egy darab email címmel, így biztosítva, hogy egy szülő csak a saját gyermekéről készült képekhez férhet hozzá.

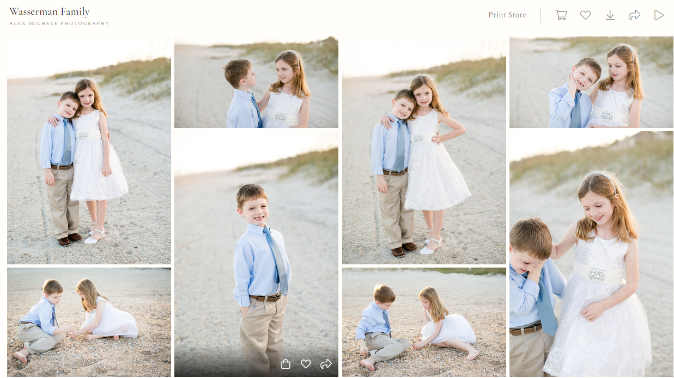
A fent említett weboldalak mindegyike egy gördülékeny, letisztult felhasználói élményt biztosít a képek megtekintéséhez. Hozzá kell tennem, hogy az alapértelmezett megjelenítés minden esetben a „kompakt lista”, még akkor is, ha a megnyitott mappa kizárólag képeket tartalmaz. Ez a megjelenítési mód nem ad lehetőséget a szülőknek az előnézeti képek alapján történő válogatásra. Azonban egy kép megnyitása után könnyen böngészhetünk közöttük.

A fenti weboldalak mind megfelelő platformként szolgálnának a fényképek *megosztásához*, azonban egyik sem oldja meg a képek *megrendelésének* a problémáját. Amennyiben erre is van szükségünk, és a felhő alapú fájlszolgáltatások mellett szeretnénk maradni, úgy használhatjuk a Google által fejlesztett Google Photost. Ez a platform kizárólag a képek megosztására és tárolására jött létre. A fotós itt is megoszthat egy mappát (avagy albumot) a szülőkkel az email címük ismeretében, akik a képeket csempézett elrendezésben is megtekinthetik. A Google Photos jelenleg fénykép nyomtatást, és vászonkép készítést vállal az ide feltöltött képekből [1], így a fényképek megrendelése is megoldható.

Ez az opció azonban nem megfelelő egy iskola fotózási környezetben, hiszen a fotósnak a képek kiküldésekor egy vízjellel ellátott, rosszabb minőségű képet célszerű kiküldenie, hogy elkerülje, hogy a szülők maguk hívassák elő a fényképeket, vagy csak digitális formában tartsák meg. Ez azért van, mert sok esetben a fényképészek nem a fotózáskor kapnak díjazást, hanem a megrendelt képekből kapnak csak jutalékot. Ezt számításba véve, a Google Photos sem megfelelő megoldás az általunk leírt célra.

Összegzésképpen, a fent említett szolgáltatások a fényképek megosztására megfelelő platformot nyújtanak, azonban azon felül, hogy a fotósnak jelentős munkát jelentenek (a fényképek elrendezésével és a jogosultságok kiosztásával), a fényképek megrendelésére sem biztosítanak megfelelő felületet. Így kijelenhetjük, hogy ezek a szolgáltatások nem fedik le ugyan azokat a funkciókat, melyekre egy iskolai fotósnak szüksége van.

## Pixieset

A Pixieset (pixieset.com) célja igen hasonló ahhoz, amit én szeretnék elérni: fényképek megosztása bárkivel, és azokat tőlünk megvásárolhatják. A weboldal segítségével bármennyi fotó feltölthető egy albumba, melyet jelszóval levédhetünk. A vásárló a weboldalunk felkeresésekor beírhatja az általunk megadott kódot, és így az összes képet megtekintheti jó minőségben, de vízjellel ellátva. A fényképésznek lehetősége van beállítani milyen szolgáltatásokat vállal, így akár a fényképek méreteit, és az ajándéktárgyakat is beállíthatja, melyekből a vásárlók rendelni tudnak. A Pixieset weboldal továbbá több fizetési átjárót is támogat (melyek közül a PayPal támogatja csak a forint pénznemet).

2.1. ábra: Képernyőkép a Pixieset weboldalról (Forrás: https://bondi.pixieset.com/wassermanfamily/)

A Pixieset így a szükséges funkciók legnagyobb részével rendelkezik. Egy fontos limitáció azonban, hogy nincs lehetőség intézmény szinten kezelni az albumokat. Minden gyermeknek külön albumot kell kézzel létrehozni, ami egy több száz fős iskolánál nem várható el a fotóstól. A fizetési határidők beállítása, és készpénzes fizetés nem megoldható egy ilyen platform használata esetén. A Pixieset tehát egy jó választás lehet például esküvőfotók megosztására, azonban iskolafotózáshoz nem megfelelő.

## Iskolaévkönyv

Hosszas kutatás után, a magyar piacon a DATATRONIC cég által fejlesztett Iskolaévkönyv (*iskolaevkony.hu)* weboldal az egyetlen olyan szolgáltatás, amely igazán hasonló feladatot lát el, mint a feladatban leírt weboldal. A weboldalról készült Youtube videó alapján [2] az oldal használata szubjektív véleményem szerint nem elég felhasználóbarát.

A képen szöveg, képernyőkép, embléma, Betűtípus látható

Automatikusan generált leírásA fényképész egy úgy nevezett „boltot” indít a weboldalon és megadja a vállalkozásának adatait. Ez után tud létrehozni osztályokat, majd tölti fel az ide tartozó fényképeket. A képek megosztása - hasonlóképpen, mint a tervezett rendszerben – jelszó segítségével működik. A megadott jelszó beírása (2.3.1. ábra) után a felhasználónak választania kell a jelszóhoz rendelt osztályok között ugyanis elképzelhető, hogy egy szülőnek több gyermeke is ját ugyan abba az iskolába).

2.2. ábra: Kód beírása az iskolaévkönyv weboldalon

Ez után megjelenik az elkészült képek előnézetének listája, vízjellel ellátva. Egy képet kiválasztva a felhasználó megnézheti azt kicsit nagyobb felbontásban, és megadhatja, hogy milyen formában szeretné azt a kosárhoz adni. Jelen pillanatban két választási lehetőség létezik csak: *Digitális* és *Nyomtatott*. (2.3.2. ábra)

A képen szöveg, képernyőkép, Webhely látható

Automatikusan generált leírásAz iskolaevkonyv.hu nem ad lehetőséget a fényképésznek, hogy ő maga végezze/végeztesse a fényképek előhívását, ajándéktárgyak készítését. Az oldal célja valójában az, hogy egy fotóst összekössön egy – a weboldal üzemeltető által kiválasztott - fénykép előhívó laborral.

2.3. ábra: Kosárba rakás az iskolaévkönyv weboldalon

## Összegzés

A fenti rendszereket vizsgálva világossá válik, hogy a kiszemelt célra még nem készült megfelelő rendszer. A fényképek megosztására több megoldást is bemutattam, azonban az iskolai fotózás esetében igen specifikus követelményeknek kell megfelelni.

A platformokat összehasonlítva (2.1. Táblázat) látszik, hogy a felhőalapú rendszerek kizárólag a fájlok megosztására alkalmasak. Ezek a rendszerek alkalmasak arra, hogy a miniatűrök kinyomtatása és elküldése helyett a fényképész digitálisan juttassa el azokat a szülőknek. Arra azonban, hogy a szülők hogyan tudnak ezek közül a képek közül választani és rendelést leadni, arra nem adnak választ.

A fent említett felhőalapú fájlszolgáltatásokon felül a Google Photos és az Iskolaévkönyv lehetőséget adnak a képek megrendelésére is, azonban azokat csak az általuk meghatározott formában lehet megrendelni. A fényképész, aki maga szeretne gondoskodni a fényképek előhívásáról így meg lesz fosztva az abból származó bevételtől.

A Pixieset az eddig említett akadályokat áthidalja, ellenben a diákokat - hasonlóan mint a fent említett rendszerekben – a fényképész köteles külön-külön kezelni, albumokat és a hozzájuk tartozó jelszavakat kézzel létrehozni. Ez a folyamat rengeteg időt venne igénybe, így beláthatjuk, hogy a Pixieset sem biztosít elég nagy eszköztárat arra, hogy ésszerű keretek között lehessen alkalmazni iskolafotózások esetén.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Felhőalapú tárhely szolgáltatások** (Google Drive, OneDrive, Dropbox) | **Google Photos** | **Pixieset** | **Iskolaévkönyv** |
| **Fényképek megosztása** | Van | Van | Van | Van |
| **Fényképek megosztása egyetlen emberről** | Van | Van | Van | Nincs |
| **Alapértelmezett megjelenítés** | Lista | Miniatűrök | Miniatűrök | Miniatűrök |
| **Automatikus vízjelezés** | Nincs | Nincs | Van | Van |
| **Rendelés leadása** | Nincs | Nincs | Van | Van |
| **Intézmények kezelése** | Nincs | Nincs | Nincs | Van |
| **Osztályok kezelése** | Nincs | Nincs | Nincs | Van |
| **Jó minőségű digitális képek vásárlása** | Nem automatizált | Nem automatizált | Van | Nem automatizált |

2.1. Táblázat: Hasonló rendszerek összehasonlítása

# Technológiák vizsgálata

## 

## Backend környezetek

A portál elkészítéséhez fontos megvizsgálnunk, hogy milyen környezetben éri meg elkészíteni a backend oldalát az alkalmazásnak. A piaci trendek, és a szóbanforgó feladat követelményeinek átgondolásával a PHP, Python és .NET rendszereket vizsgáltam meg mélyebben, hogy átfogó képet kapjak a lehetőségekről, és a megadott célra a legjobb alapot választhassam.

### PHP

A PHP egy nyílt forráskódú, ingyenes script nyelv, ami a szakdolgozat írásakor a weboldalak több mint 76%-ának az alapját képezi [3], tehát 5 weboldalból majdnem 4 használja ezt a technológiát, többek között a Facebook, Wikipedia és a Tumblr [4]. A PHP előnyeinek és hátrányainak megvizsgálásához röviden szeretném ismertetni a létrejöttének és elterjedésének történelmét.

A PHP-t 1994-ben alkotta meg Rasmus Lerdorf azzal a céllal, hogy tudja követni, hogy ki nyitja meg az önéletrajzát. [5] Ez után 1995-ben adta ki az első hivatalos, open source verzióját. A PHP programozási nyelv azóta rengeteg változáson ment keresztül, jelen pillanatban a legfrissebb verzió a PHP 8.2. A népszerűségét több dolognak is köszönheti, melyek közül először a nyelv platform-függetlenségét emelném ki. Ez az a tényező, ami lehetővé teszi, hogy a fejlesztők az operációs rendszerüktől függetlenül képesek legyenek PHP-ban fejleszteni. Valamint ennek köszönhetően az elkészült webalkalmazás majdnem bármilyen szerveren telepíthető.

A platformfüggetlenségből eredően sok Content Managment Service (CMS) mint például a WordPress és a Drupal is PHP-ban lettek írva. Manapság web szerver bérlésekor is majdnem garantált, hogy ezeket a CMS-eket támogatni fogja a webszerver szolgáltató (tarhely.eu, dima.hu), tehát ezek a szerverek is PHP telepítéssel kerülnek átadásra az ügyfélnek.

A PHP egy másik nagy előnye, hogy elterjedtségének köszönhetően kifejezetten nagy közösség vesz részt a fejlesztésében. Ebből adódóan az esetleges sérülékenységek is hamar javításra kerülnek. Ezen felül sok előre megírt könyvtár és keretrendszer (Laravel, CodeIgniter, CakePHP) jött létre PHP nyelven, melyek segíthetik a fejlesztést.

Ezeket az előnyöket szem előtt tartva szeretném megvizsgálni a nyelv hátrányait is. A korának, és annak köszönhetően, hogy nyílt forráskódú, a PHP beépített nyelvi elemei helyenként nagyon inkonzisztensek. A függvények nevei sok esetben nem követnek semmi féle logikai elvet [6], így a fejlesztés során sokszor a dokumentációra kell hagyatkoznunk.

Egy másik jelentős hátrány a PHP használata során az az, hogy a nyelv nem garantál visszafelé kompatibilitást frissítések esetén [7]. Könnyen előfordulhat tehát az, hogy a szerveren futó PHP verzió frissítésekor a megírt PHP scriptek többé nem működnek rendeltetésszerűen. Ebből adódóan a karbantartás ideje jelentősen megnőhet egy olyan rendszer esetén, mint amilyet én tervezek létrehozni.

Egy másik szempont, amit érdemes még szem előtt tartani az az a tény, hogy a PHP egy értelmezett nyelv. Az évek során a PHP verziók jelentős sebességnövekedést értek el [8], azonban az ASP.NET-hez mérten (ami általában C# nyelvet használ) jelentősen lassabbnak számít. [9]. A tervezett iskolai fotó megosztó platformon azonban ezek a sebességkülönbségek nem jelentenek túl nagy problémát, hiszen a weboldal célja a képek megosztása, így a szűk keresztmetszetet előreláthatóan a fényképeket tároló szerver és a felhasználó közötti internetkapcsolat sebessége fogja jelenteni.

Összefoglalásképp, a PHP egy jó választás lehetne a probléma megoldására. A legfontosabb ok, ami miatt úgy döntöttem, hogy más nyelvet választok a feladat megoldására, az a PHP nehezen karbantarthatósága. A portál, amit ki szeretnék építeni, az egy összetettebb rendszer, amelynél fontos, hogy a jövőben kiadott környezet frissítések ne rontsák el a működését.

### Python

A Python egy ingyenes, nyílt forráskódú programnyelv, melynek születése még a fentebb említett PHP nyelv létrejöttét is megelőzte, hiszen az első nyilvánosságra hozott verziója (0.9.0) 1991 februárjában volt. A megalkotója Guido van Rossum holland programozó volt, aki egy olyan programozási nyelv kifejlesztését tűzte ki céljául, melyet könnyű olvasni, írni és karbantartani. [10]

A kód olvashatóságát nagyban segíti, hogy a nyelv szigorúan előírja azt, hogy melyik sornak mennyivel kell beljebb kezdődnie (indentáció), így könnyen követhetővé válnak az összetartozó kódrészletek.

Megjelenése után nagy sikerre tett szert annak köszönhetően, hogy a Python nyelven írt kód könnyen olvasható, és a szintaxisát igen könnyű elsajátítani. A nyelv sok olyan sajátossággal bír, amely azt a célt szolgálja, hogy egy adott algoritmust a lehető legkevesebb sor kód megírásával implementáljunk (pld listák bejárásakor könnyen válogathatjuk ki hátulról indulva a listában található minden harmadik elemet). A programnyelv az átfogó standard könyvtárának köszönhetően több programozási paradigmát is meg tud valósítani, tehát használható objektum orientált vagy funkcionális programozás alkalmazására. [11]

A nyelv népszerűségének köszönhetően a mai napig újabb és újabb könyvtárak és keretrendszerek jelennek meg a Python programnyelvhez. Kutatásom során a backend programozásra alkalmas Python alapú keretrendszerek közül a Django-t (2005) és a Flask-et (2010) vizsgáltam.

A Django egy kiforrott, viszonylag nehezen elsajátítható keretrendszer, melynek elsődleges célja egy full-stack rendszer kiépítése. A Model-View-Template (MVT) architektúrát követi, melyben a megjelenítési réteget két külön egységre bontja, így lehetővé téve azt, hogy a megjelenítés megtervezésében olyan ember is részt tudjon venni, aki nem ért a programozáshoz. A webportál elkészítésekor nem lesz szükség ezekre a funkciókra, így ezek kevés előnyt jelentenek a számomra. [12]

A Flask ezzel szemben egy „lightweight” keretrendszer, amely telepítéskor még az adatbázisok kezelését sem támogatja, erre a funkcióra bővítményeket lehet beszerezni, mint a Flask SQLAlchemy. Abból adódóan, hogy a Flask ilyen kevés funkciót tartalmaz alapból, így teljes rugalmasságot biztosít a fejlesztőnek abban, hogy hogyan építse föl, strukturálja és bővítse a kódbázist. A beépített funkciók hiány azonban egy nagyobb rendszer kiépítésekor – mint a szóban forgó iskola fotózást segítő alkalmazás – jelentősen több munkát tud jelenteni a fejlesztőnek.

### .NET

A .NET Framework egy nyílt forráskódú szoftver keretrendszer, melyet a Microsoft adott ki 2002-ben arra a célra, hogy versenybe szálljon az akkoriban egyre inkább terjedő Java programnyelvvel. A Microsoft a .NET Framework-öt már az elejétől kezdve azzal a reménnyel fejlesztette, hogy egy nap majd több operációs rendszeren fog futni, ellenben eleinte a .NET-ben írt programok kizárólag a Microsoft saját fejlesztésű operációs rendszerén működött, a Windowson. [13]

Amikor a .NET Framework nyílt forráskódúvá vált Miguel de Icaza és Nat Friedman sikeresen elérték, hogy az akár Linuxon is képes legyen futni. 2009-re már olyan megoldások is születtek (pld Mono), melyek képesek voltak akár Linuxon is működni. Ezek azonban mind körülményes megoldások voltak ugyan arra a problémára. [13]

Végül a választ a problémára a .NET Core megjelenése jelentette 2016-ban, amely a .NET Frameworkhoz hasonlóan nyílt forráskódú, azonban platform független és még gyorsabb is az elődjénél. [13] A keretrendszerek történelmét azért volt fontos megismerni, hiszen a .NET Core már lehetővé teszi, hogy a back-end szervert bármilyen operációs rendszeren futtatni lehessen.

Az ASP.NET keretrendszer egy olyan nyílt forráskódú keretrendszer, ami a .NET platformra épül, és lehetővé teszi, hogy a beérkező kéréseket C# vagy F# nyelven írt programmal dolgozzuk fel. Az első verzióját 2002-ben adta ki a Microsoft a .NET keretrendszer részeként. [14] Mára az ASP.NET a weboldalak 6,8%-ának az alapját képezi, tehát ez a második legelterjedtebb technológia a PHP után. [3]

A Microsoft 2016-ban adta ki az ASP.NET újraírt verzióját .NET Core alapra, az ASP.NET Core-t azzal a céllal, hogy az átvegye az ASP.NET helyét. Az új rendszer így ki tudja használni a .NET Core-ral járó sebesség növekedést, valamint lehetővé teszi, hogy az ne csak Windows operációs rendszereken, hanem MacOS-en és Linux-on is futtatható legyen.

A keretrendszer rengeteg beépített funkcióval érkezik, például rendelkezik a saját templating szintaxisával is, a Blazorrel, amely WebAssembly (wasm) segítségével lehetővé teszi, hogy dinamikus weboldalakat készítsünk. Így lehetőségünk nyílik .NET-ben lefejlesztett kódot futtatni a böngészőkben is. Eleinte azt is mérlegeltem, hogy a különálló frontend helyett a teljes weboldalt így hozom létre, azonban a Blazor WebAssembly megoldását nem ítéltem elég elterjedtnek ahhoz, hogy egy hosszútávú projektet ezzel az eszközzel készítsek el.

Ezen felül az ASP.NET Core számtalan olyan beépített funkcióval elérhető, melyek jelentősen fel tudják gyorsítani egy back-end alkalmazás fejlesztését, így API alapú fejlesztésre is kiváló választás lehet. Ezek közül a funkciók közül én is sokat igénybe vettem, melyeket a szakdolgozat további részében fogok említeni.

### Összefoglalás

Bár a statisztikák kimutatják, hogy a PHP uralja a weboldalak jelentős százalékát, ez a tény megmagyarázható azzal, hogy a legelterjedtebb CMS-ek, mint a WordPress is ezt a nyelvet használják. A PHP-ban megírt keretrendszerek, mint a Laravel vagy a CodeIgniter alkalmasak lennének a projekt véghezviteléhez, valamint már mind a kettő keretrendszerrel igen sok tapasztalatom van. Azonban a PHP gyakori frissítései melyek a weboldal állandó karbantartását igénylik, valamint az inkonzisztens nyelvi elemek meggyőztek, hogy ne mellettük döntsek.

A Python keretrendszerek közül is mindkettő megfelelő lenne a probléma megoldására, azonban a statisztikák szerint a weboldalak kisebb százaléka használja csak, és értelmezett nyelv létéből adódóan lassabb végrehajtást biztosít, mint a .NET Core. Az utóbbi így majdnem minden szempontból a legjobb választás a tárgyalt cél eléréséhez, valamint az egyetemi éveim során sok tudásra tettem szert a különböző nyelvi elemek és beépített eszközök körében. Tehát az iskolai fotózást támogató portálom back-end rendszerét ebben a keretrendszerben fogom elkészíteni.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Környezetek** | **PHP** | | **Python** | | **.NET** |
| **Nyílt forráskódú** | Igen | | Igen | | Igen |
| **Ingyenes** | Igen | | Igen | | Igen |
| **Nyelv típusa** | Értelmezett nyelv | | Értelmezett nyelv | | Fordított nyelv |
| **Szintaxis** | Néhol logikátlan [6] | | Nagyon könnyű | | Könnyű |
| **Használati eloszlás** [3] | 76% | | 1.4% | | 6.8% |
| **Vizsgált keretrendszerek** | **Laravel** | **CodeIgniter** | **Django** | **Flask** | **ASP.NET Core** |
| **Gyorsaság** | Közepes | Közepes | Közepes | Közepes | Jó |
| **Fejlesztést segítő beépített funkciók** | Sok | Kevés | Sok | Kevés | Nagyon sok |
| **Saját tapasztalatom** | Sok | Sok | Kevés | Nincs | Sok |

3.1. Táblázat Back-end környezetek és keretrendszerek összehasonlítása

## Frontend környezetek

A rendszer frontend oldalának megtervezésekor a Javascript keretrendszereket részesítettem előnyben. A Statista felmérései szerint a Javascript a programozók körtében a leghasználtabb programozási nyelv, a megkérdezett programozók több mint 63%-a nyilatkozott úgy, hogy használja [15]. Igaz, hogy manapság egyre nagyobb népszerűségnek örvend a WebAssembly (wasm), azonban ezek a megoldások ritkán szolgálják azt a célt, hogy a felhasználói felület működését segítsék, sokkal inkább magas erőforrású, teljesítmény-kritikus megoldások leprogramozására alkalmasak. Ez azzal magyarázható, hogy a Javascripttel szemben (ami egy értelmezett nyelv), a wasm egy már fordított kódot futtat, így nagyobb teljesítményt képes elérni komplex számítások esetén.

### Angular

Az Angular egy nyílt forráskódú, ingyenes frontend keretrendszer. Létrejötte két Google alkalmazott, Misko Havery és Adam Abrons nevéhez köthető. Az első hivatalos kiadása 2010-ben látott napvilágot AngularJS néven, amikor a legelterjedtebb módja a DOM manipulálásának még a JQuery könyvtár volt, ami lehetővé tette, hogy a weboldalak elemeit dinamikusan módosítsuk a teljes weboldal újrarenderelése nélkül. Az AngularJS ezt a működést egy teljesen másik megközelítéssel éri el. A keretrendszer segítségével Single Page Application-öket (SPA) lehet létrehozni, ahol HTML attribútumok segítségével képes leírni a fejlesztő az egyes elemek viselkedését, így ezzel megkönnyítve a munkáját. Továbbá az Angular a kétirányú adatkötést is támogatta, azaz lehetővé tette, hogy a felhasználó áltat végzett módosítások a háttérben futó JavaScript kódban is megjelenjenek, vagy épp fordítva.

Az AngularJS már megjelenésekor nagy sikert aratott, valószínűleg nem véletlen egybeesés, hogy a Facebook is pont 2010-ben kezdett el dolgozni a saját keretrendszerén, a React-ten. [16] A két keretrendszer versengése során a Google sok módosítást végzett a saját fejlesztésű keretrendszerén, 2016-ban ki is adta a rendszer teljesen újraírt változatát, ekkor már csak Angular néven, ami JavaScript helyett TypeScriptet használ.

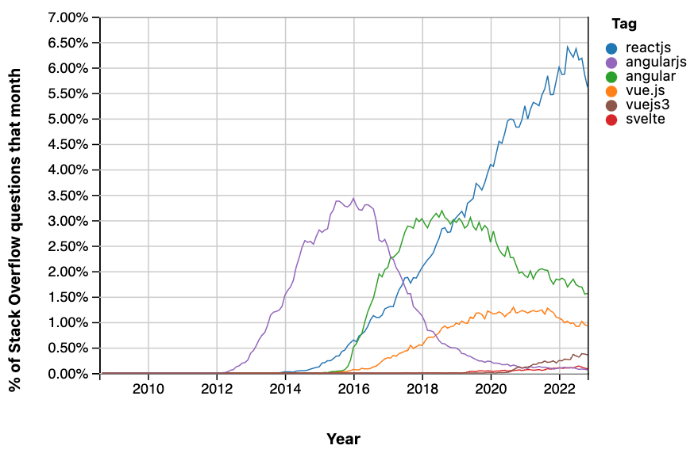
### Vue.js

A Vue.js egy ingyenes, nyílt forráskódú Javascript könyvtár SPA weboldalak készítésére. Megalkotása Evan You nevéhez kötődik, akiről érdekesség képpen megemlíteném, hogy az ő nevéhez köthető továbbá a Vite megalkotása is, egy széleskörben elterjedt Javascript buildelési eszköz. A Vue.js első hivatalos megjelenése 2014 februárja volt, így ez a legfiatalabb frontend keretrendszer a szakdolgozatban megvizsgáltak közül. 2013-ban, amikor You a Google alkalmazottja volt, szüksége volt egy olyan keretrendszerre, amivel gyorsan és hatékonyan tudja létrehozni nagyobb méretű felhasználói felületek prototípusát. Erre a célra akkoriban az Angular és a React nem nyújtott megfelelő megoldást. [17]

A keretrendszer megalkotásakor nagy figyelmet fordítottak arra, hogy az egy kis méretű, letisztult megoldás legyen. Ebből kifolyólag az elért teljesítmény, és memóriafelhasználás sok esetben jobb eredményt ér el, mint versenytársai. A Vue.js ma már megfelelő rendszer lenne egy nagyobb webalkalmazás lefejlesztésére is, azonban annak köszönhetően, hogy igyekeztek minél kevesebb fölösleges funkciót belerakni, a megírt kódbázis bővítése körülményesebb, több külső könyvtár telepítése szükséges, mint például a React vagy az Angular esetében.

Kutatásaim során arra jutottam, hogy a Vue.js egy jó választás lenne egy kisebb rendszer kiépítése esetén, de egy ekkora méretű projekthez túl nagy tervezést igényelne a használata.

### React

A React a fentiekhez hasonlóan egy ingyenes, nyílt forráskódú frontend keretrendszer, melyet a Facebook (avagy Meta) fejleszt. Az első hivatalos megjelenése 2013-ban volt. A React megközelítése az SPA weboldalt komponensekre bontja, valamint az Angular és Vue.js keretrendszerekkel szemben egyirányú adatkötésekre támaszkodik. A React továbbá bevezette a Virtual DOM (VDOM) fogalmát is a saját rendszerébe. A VDOM (ebben az esetben a ReactDOM) alapja, hogy a felhasználói felület egy virtuális reprezentációját is a memóriában tartja. Ennek a megközelítésnek a segítségével minden módosítás, amit a virtuálisan eltárolt elemeken végzünk, az megjelenik a felhasználónak megjelenített DOM-on is, anékül, hogy ezzel nekünk foglalkoznunk kéne. Ez nagy előnyt jelent a korábbi fejlesztést segítő könyvtárakhoz képest (mint a JQuery), ahol minden módosítás egyből a DOM-on lett végrehajtva. A VDOM-on végzett módosítások gyorsan elvégezhetőek, és a valódi DOM-ra csak akkor képződnek le a módosítások, amikor azok valójában szükségesek. „Ez absztraktálja az attribútum manipulációt, esemény kezelést és manuális DOM frissítést, amit egyébként egy alkalmazás fejlesztésekor nekünk kéne elvégeznünk”. [18]

3.1. ábra: A Stack Overflow weboldalon feltett kérdések száma frontend technológiák szerint [19]

Megjelenésekor az Angular nagyobb webalkalmazások futtatásakor nagy teljesítménycsökkenést idézett elő, így a React, ahol ugyan ez a probléma nem állt fent, nagy népszerűségre tett szert. Mára, a Stack Overflow weboldal felmérése szerint a React-tel kapcsolatos kérdések száma több mint kétszerese az Angular-rel kapcsolatos kérdésekének. (3.2.1. ábra: A Stack Overflow weboldalon feltett kérdések száma frontend technológiák szerint [19])

A React bővebb ismertetését egy későbbi fejezetben fogom leírni, amit azonban előre ki szeretnék emelni, az az, hogy a React – bár még támogatja – de nem ajánlja a komponensek létrehozását osztályokként. Ehelyett mindenhol a függvény komponensek alkalmazását ajánlja, melyek képesek valamilyen bemenetre (properties), a saját állapotuk felhasználásával (state) valamilyen HTML-re lefordítható eredménnyel visszatérni (JSX). Ez a viselkedés az, ami jelentősen megkönnyíti a tesztelés folyamatát, hiszen tudjuk, hogy milyen bemenetre milyen kimenettel kell visszatérnie a programnak.

A React továbbá támogatja a TypeScriptet is, így lehetőségünk van fejlesztés során kiszűrni azokat a hibákat, melyeket JavaScriptben nem vennénk észre. A keretrendszer alkalmas nagyobb projektek kivitelezésére, és azok hosszú-távú támogatására is. Ezen szempontok megvizsgálása után így úgy döntöttem, hogy számomra a legmegfelelőbb környezetet a React nyújtja majd a frontend létrehozására.

### Összefoglalás

A vizsgált frontend keretrendszerek közül mind a három megfelelő lenne a kitűzött cél eléréséhez. A Vue.js-t abból az okból vetettem el, mert a bővíthetőségre nem fejtet elegendő hangsúlyt a másik kettő lehetőséghez képest. A React és az Angular között végül azért döntöttem az előbbi mellett, mert a felhasználásával már sok tapasztalatom van nagyobb méretű frontend SPA weboldalak készítésében, és úgy érzem birtokában vagyok annak a tudásnak, ami lehetővé teszi azt, hogy a portált olyan minőséggel elő tudjam állítani, mint amilyen elvárást magammal szemben támasztottam.

|  |  |  |  |
| --- | --- | --- | --- |
| **Környezetek** | **Angular** | **Vue.js** | **React** |
| **Nyílt forráskódú** | Igen | Igen | Igen |
| **Ingyenes** | Igen | Igen | Igen |
| **TypeScript** | Beépített | Támogatott | Támogatott |
| **Bővíthetőség** | Nagyon jó | Közepes | Nagyon jó |
| **Típus** | Objektum orientált | Objektum orientált | Funkcionális |
| **Saját tapasztalatom** | Kevés | Nagyon kevés | Sok |

3.2. Táblázat: Frontendes keretrendszerek összehasonlítása

# Használt technológiák ismertetése

## Backend technológiák

### SimplePay

## Frontend technológiák

### TypeScript

### ESLint

### React

Ebben az alfejezetben azt szeretném ismertetni, hogy a React keretrendszer milyen funkcióira fogok támaszkodni a projekt kivitelezése során.

#### JSX

A React egyik sajátossága, hogy lehetőséget biztosít arra, hogy HTML kódot írjunk a JavaScript kódunkban. Ezt a JSX (JavaScript XML) szintaxissal tudjuk elérni, amire a React dokumentációja szerint [20] azért volt szükség, mert a keretrendszerben a renderelés logikája kéz-a-kézben jár a UI logikájával. Ez a működés teszi lehetővé azt, hogy függvények visszatérési értéke valójában HTML kód legyen, így a függvény komponenst alkosson. A JSX ránézésre nagyon hasonló a HTML-re, azonban a megszokott HTML tag-ek helyett (div, p, nav, stb) lehetőségünk van a saját komponenseink felhasználására is.

#### Komponensek

A képen szöveg, képernyőkép, Betűtípus, diagram látható

Automatikusan generált leírásMint azt már egy korábbi fejezetben is leírtam, a React-ben eleinte a komponensek kizárólag osztályokként voltak jelen. Ezeknek az osztályoknak egy előre definiált életciklusuk van, melyek egy adott sorrendben vannak meghívva akárhányszor a komponens megjelenítésre kerül. (4.2.1. ábra) Ezen függvények segítségével tudja a komponens beállítani a saját state-jét a prop-jai felhasználásával, és a render metódussal tud visszatérni a megjelenítendő JSX kóddal. Ezt a megközelítést még ma is támogatja a React, és még nem is tettek bejelentést, hogy valaha vissza szeretnék vonni. Azonban a dokumentáció szerint mára kizárólag a funkcionális komponensek használatát ajánlják, [21] melyek olvashatóbb és tesztelhetőbb kódot eredményeznek, így csökkentve a hibák előfordulásának lehetőségét. Ezen kívül gyorsabb végrehajtást, és kisebb fájlméretet is eredményeznek. A továbbiakban minden komponenssekkel kapcsolatos általánosítás a függvény komponensekre vonatkozik.

4.1. ábra: React osztálykomponensek (forrás: https://medium.com/@tgholami/react-function-components-lifecycle-8a6ede4b8c2e)

#### State management, React hook-ok

A Reactben a komponens egy olyan függvény, amely a JSX szintaxis felhasználásával tulajdonságokat (properties) kap meg a szülőjétől, és nyilvántartja a saját jelenlegi állapotát (state). Az osztálykomponensek esetén a state nyilvántartása egy magától értetődő folyamat volt egy objektum orientált programozónak, hiszen a komponens state-je az osztály state változójában van eltárolva. Azonban amikor a React áttért a függvény komponensek használatára, akkor egy új megközelítést kell alkalmaznunk. Ahhoz, hogy ezek a komponensek ugyanazokat a funkciókat legyenek képesek ellátni, mint elődjük, ahhoz egy új fogalmat kellett bevezetniük: a hook-okat.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírásA hook az egy olyan függvény, ami hozzáférést ad a fejlesztőnek a React bizonyos funkcióihoz. A hook-ok neve általában konvenciót követve a „use” angol szóval kezdődik. A komponens state-jének kezelésére például a useState hook alkalmas. Ennek a függvénynek egy opcionális paramétere van: az adott state változó kezdeti értéke, a visszatérési értéke pedig egy két elemű tömb, melynek elemeit könnyen kiolvashatjuk a JavaScript tömb destrukturálás felhasználásával. A tömb első eleme a state jelenlegi értéke, a második pedig az a függvény, amivel azt módosítani lehet. (4.2.2. ábra)

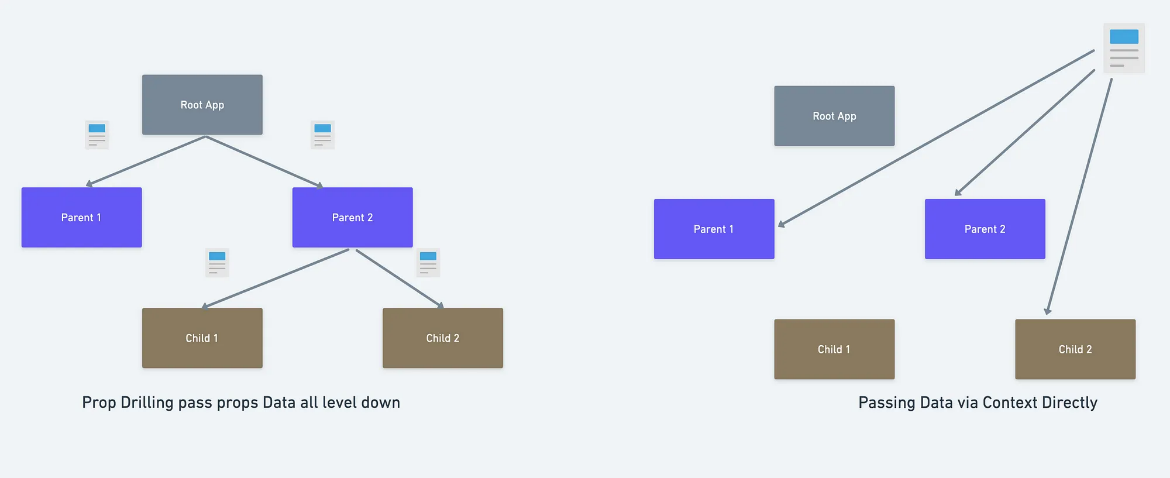
4.2. ábra: Példa a useState hook használatára (forrás: saját ábra)

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírásEgy másik nagyon fontos hook aminek használata elengedhetetlen a fejlesztés során, az a useEffect. Ez a függvény teszi lehetővé azt, hogy egy olyan „mellékhatást” tudjunk specifikálni, aminek a komponens renderelésekor lefutnia. Az első paramétere maga az effect, tehát az a callback függvény, ami leírja minek kell történnie. A második paraméter az az úgy nevezett függőségi tömb (dependency array). Ez a tömb specifikálja, hogy milyen értékek megváltozásakor kell lefutnia az effect-nek. Amennyiben a tömb üres, úgy az adott kód minden újrarendereléskor lefut. A hook visszatérési értéke egy olyan opcionális függvény lehet, aminek az a célja, hogy a komponens „feltakarítson maga után”, tehát például leiratkozzon olyan eseményekről, amikre az effect meghívásakor feliratkozott. A useEffect tehát jó megoldás arra az esetre, ha egy API hívást kell végrehajtanunk amikor a komponens megjelenik, vagy ha fel kell iratkoznunk a böngésző eseményeire.

4.3. ábra: Példa a useEffect hook használatára (forrás: saját ábra)

Az utolsó hook amit ki szeretnék emelni, az a useContext. Enélkül a hook nélkül, amikor kettő vagy több komponens ugyan azt a state-et kell hogy használja, olyankor a kézenfekvő megoldás az lenne, hogy a komponensek egyik közös szülőjébe kiemeljük ezt a state-et, és azt a komponenseknek property-ként adjuk át (lifting state). Ez a megközelítés teljesen helytálló addig, amíg pár szorosan összekapcsolt, a VDOM hierarchiában egymáshoz közel álló komponensről beszélünk. Azonban amikor ezek a komponensek egymástól teljesen elszigetelt helyen jelennek meg, olyankor a közös state-et rengeteg köztes komponensen keresztül kell továbbadnunk míg meg nem érkezik ahhoz, ami valójában használni is fogja. Ezt a jelenséget nevezzük prop drillingnek, ami egy olyan kódot eredményez, ahol egy kisebb módosítás is nagy munkával járhat, könnyen hibát véthetük a komponensek felparaméterezésével, és egy nehezebben olvasható kódot is kapunk. Erre a problémára ad megoldást a useContext hook.

Használatához szükségünk van arra, hogy komponenseknek egy szülőkomponensében létrehozzuk a kontextust. A state változtatások a továbbiakban is ebben a komponensben fognak megtörténni, azonban ahelyett, hogy azt property-ként adná át a komponenseinek, inkább létrehoz egy kontextust a createContext függvénnyel, majd ezt felhasználva az összes komponenst ahol a state-re szükség lesz, egy úgynevezett ContextProvider komponensbe helyezi, és ennek a komponensnek adja csak át prop-ként a kívánt értékeket. A gyermek komponensek innentől bármikor elérik a kontextus jelenlegi értékét, anélkül hogy szükség lenne kézzel megírni azt, hogy az az adat hogyan jut el hozzájuk.

4.4. ábra: Prop drilling és kontextus összehasonlítása (forrás: https://medium.com/geekculture/props-drilling-v-s-context-api-which-one-is-the-best-75c503d21a65)

#### Saját hook-ok

A keretrendszer továbbá azt is lehetővé teszi, hogy saját hook-okat definiáljunk, ami valójában olyan komponens, ami nem egy JSX elemmel tér vissza, de mégis rendelkezhet property-kel és state-tel. Saját hook-ok segítségével lehetőségünk van az üzleti logikát kiszervezni a komponensekből, ami olvashatóbb kódot eredményez, hiszen a hook nevéből ki tudjuk találni, hogy annak mi a célja – anélkül, hogy tudnunk kéne hogyan működik.

#### Create-react-app

TODO

### Zustand

A portál működéséhez több helyen is létre kell hoznom olyan state-et, melyet több komponens is használni fog, mint például a kosár tartalma, a jelenleg beírt rendelési kód, valamint az admin felületen a jelenleg bejelentkezett felhasználó.

Mint ahogyan azt a 4.2.3 fejezetben is leírtam, egy ilyen jellegű state kezelésére a useState nem jó megoldás, hiszen prop drillinget idézne elő. Ennek elkerülése végett használhatjuk a React context-et is és a useReducer hookot, azonban a tudásom bővítése céljából úgy döntöttem, hogy egy state manager könyvtár alkalmazását választom a fejlesztéshez.

A state managerek általánosságban ugyan azokat az alapelveket követik. Egy közös tulajdonságuk, hogy a state módosítása nem közvetlenül történik – azaz a komponensek nem a state-ben tárolt változók értékeit módosítják, hanem egy olyan függvényt hívnak meg, ami módosítja azt. Ez a működés segíti a hibakeresést, és a tesztelhetőséget, hiszen a state változását és a felhasználó interakcióit könnyen nyomon tudjuk követni a meghívott függvények nyilvántartásával. A másik közös vonás a state manager-ekben az az, hogy a state-et nem a komponensekben tároljuk, hanem egy központi helyen, melyet store-nak nevezünk.

Kutatásaim során a state managerek közül a Redux-ot, és a Zustand-ot hasonlítottam össze. A Redux a legelterjedtebb state manager könyvtár jelenleg [22], azonban a használata jelentősen le tudja lassítani a fejlesztési folyamatot, hiszen a megvalósítása sok kódolással jár. Használatát leginkább a nagy méretű alkalmazások esetében ajánlják. A használatához sok szabályt kell betartanunk ahhoz, hogy az egy bárki által átlátható és karbantartható kódot eredményezzen.

A Zustand ezzel szemben egy „lightweight” könyvtárnak mondható, melyet könnyű megtanulni és használni is. A használatát nem ajánlják óriási projektek esetén, azonban én úgy ítéltem, hogy a fotó megrendelő portál mérete nem lépi át ezt a küszöböt.

### Axios

A frontend és a backend alkalmazás közötti kommunikáció megvalósításához két megoldást vizsgáltam meg. A Javascriptbe beépített fetch() függvény egy átfogó megoldást nyújt minden http kérés lebonyolításához. A fetch függvény az promise-based - azaz a visszatérési értéke a generikus Promise objektum – így a hibakezelés megvalósítása és művelet végzés az eredménnyel nagyon egyszerű.

Az Axios könyvtárnak azonban van pár olyan funkciója, ami miatt előnyben részesítem az alapból beépített fetch-csel szemben. Az első nagy előnye, hogy akár régebbi böngészőkben is képes kéréseket küldeni. A fetch API csak 2015-ben jelent meg [23] így az azelőtt megjelent böngészőket használó szülők/nagyszülők nem tudnák használni a weboldalt. Egy másik funkciója melyre szükségem lesz a fejlesztés során, az a HTTP interceptors. Ennek a funkciónak a felhasználásával módosítani tudjuk a kérések header-jét vagy tartalmát még azelőtt, hogy elküldenénk azt. Ez lehetővé teszi, hogy az admin oldalon az autentikáció után a JWT token minden request header-jébe bekerüljön anélkül, hogy erre figyelni kéne az API-al való kommunikáció során. Ezek alapján úgy döntöttem, hogy a beépített fetch API helyett az Axiost fogom használni.

### MUI (Material UI)

A weboldal megjelenésével kapcsolatban nem kellett sokat gondolkoznom. A célom az, hogy egy könnyen használható weboldalt hozzak létre, melyet a nem hozzáértő felhasználók is kezelni tudnak. Erre egy letisztult, minimalista weboldalt képzeltem el, melynek a megvalósításához a MUI azaz a Material UI egy tökéletes eszköz. Ez a könyvtár számtalan előre megírt komponenst szolgáltat a fejlesztéshez. Ezeknek a komponenseknek a megjelenése már alapból megfelelő lenne, de a MUI lehetőséget biztosít arra, hogy saját témát definiáljunk, így személyre szabva a weboldal által használt színeket, betűméreteket és hasonló vizuális tulajdonságokat.

### React Final Form és mui-rff

Mint minden online bolt esetében, nekünk is szükségünk lesz adatok bekérésére a felhasználótól. Véleményem szerint a felhasználói élménynél sokat számít az, hogy az adatok bevitelénél hogyan, és mikor jelennek meg a hibaüzenetek az esetleges elgépelések esetén. A felhasználótól bekért adatok sok esetben lehetnek rejtett hibaforrások, így célszerű amennyire lehet leszűkíteni a lehetségesen begépelt adatok halmazát, mind például egy email cím esetében nem szabad engedünk, hogy a vásárló érvénytelen formátumú címet adjon meg, hiszen így elveszítenénk a lehetőséget a közvetlen kapcsolattartásra a rendelés leadása után.

Tehát egy felhasználóbarát, de mégis kiszámíthatóan működő weboldal esetében elengedhetetlen a kliens oldali adat validációja. Ezt a validációt nem bonyolult megírni külön-külön minden bekért mező esetén, azonban a React Final Form könyvtár használatával nincs szükség rá, hogy ezzel foglalkozzunk, hiszen sok segítő funkcióval rendelkezik. Képes a beviteli mezők validációjára, a hibák megjelenítésére, valamint nem engedi továbblépni a felhasználót amíg az összes hiba nincs kijavítva.

A React Final Form alapvetően nem a MUI használatára lett tervezve, azonban a mui-rff nyílt forráskódú könyvtár orvosolja ezt a problémát, és olyan beviteli mezőket tesz elérhetővé, amik egyszerre használják a MUI által definiált témát és a validációs hibaüzeneteket is jól jelenítik meg. A felhasználótól bekért adatok kezelésére tehát ennek a két könyvtárnak a funkcióit fogom használni a fejlesztés során.

# Tervezés

## Rendszerterv

## A képen szöveg, képernyőkép, Betűtípus, diagram látható Automatikusan generált leírás

5.1. ábra: A back- és frontend kommunikációja a vásárlás során

## Funkciólista

### Adminisztrátori funkciók

* **Fényképészek kezelése.** Az adminisztrátor új fényképészeket tud létrehozni a rendszerben, valamint törölni őket.

### Fényképészi funkciók

* **Intézmény kezelés.** A fényképészeknek lehetőséget kell biztosítani arra, hogy új intézményt tudjanak létrehozni a rendszerben, valamint azokat módosítani tudják. Az intézményhez tartozó határidők, elérhető fizetési módok itt módosíthatóak. Egy intézmény egyszerre csak egy fényképészhez tartozhat. Elvárás továbbá, hogy intézményenként külön-külön specifikus üzenet jelenjen meg a képek fölött a kilistázáskor, így ennek a megadására is lehetőséget kell biztosítani egy WYSIWYG szövegszerkesztő segítségével.
* **Osztályok/csoportok kezelése.** Az egyes intézményeknél lehetőséget kell biztosítani arra, hogy a fényképész meg tudja adni az adott intézményhez tartozó osztályokat vagy csoportokat.
* **Szolgáltatások kezelése.** Az fényképészek megadhatják milyen szolgáltatások rendelhetők az általuk készített fényképekből.
* **Csomagok kezelése.** A fényképész által készített képekből csomagokban is lehetőség van a rendelésre. Például létrehozhat egy „karácsonyi csomag” nevű csomagot, amiben 1 hógömb, és 6 darab 10x16-os kép található. A vásárló amikor ezt a csomagot választja, akkor tetszőleges képeket válogathat bele a csomagba.
* **Általános beállítások kezelése.** A fotósnak több további beállítási lehetőséget is biztosítani kell. Többek között a kiküldött email-ek szövegét, a postai kiszállítás árát és a minimum rendelési összeget is, ami felett a vásárló megkapja az összes képet digitálisan.
* **Képek feltöltése.** A fényképész ki tud választani egy intézményt, és oda fel tudja tölteni az elkészített képeket tartalmazó mappákat, valamint kezelni tudja a már feltöltött mappákat. A képekre feltöltéskor vízjel kerül.
* **Rendelések kezelése.** A fényképésznek látnia kell az összes rendelést, amit a képeire adtak le, a vásárló adatait módosítani kell, hogy tudja, valamint azok státuszát is megváltoztathatja. A rendelések között szűrési lehetőséget kell biztosítani.
* **Rendelői összefoglaló generálása.** A rendelésekből egy összefoglaló pdf-et kell összeállítani, ami tartalmazza a rendelések alapvető információit (név, osztály, fizetendő összeg, stb.) és tartalmát.
* **Statisztika megtekintése.** TODO valami grafikon?

### Vásárlói funkciók

* **Rendelési kód beírása.** A fényképésztől kapott rendelési kód helyes beírása után a vásárló átirányításra kerül az ahhoz tartozó képek listájához.
* **Válogatás a képek között.** A kód beírása után az összes kép miniatűrje kilistázásra kerül. A vásárló a képre kattintva a kép rendelési felületre kerül. A képeket meg is lehet jelölni egy csillaggal, hogy könnyű legyen visszatérni rá. (TODO)
* **Kép rendelési felület**. A képet itt lehet nagyobb felbontásban megtekinteni, valamint (amennyiben a rendelési határidő még nem telt le) ki lehet választani, hogy milyen formában kívánja a felhasználó megvásárolni a képet. A választható csomagok, papírképek és ajándéktárgyak itt kerülnek kilistázásra. A mennyiség megadása után a termékek a kosárba kerülnek.
* **Kosár.** Az összes eddig kiválasztott termék listája itt található. A termékek mennyiségét itt módosíthatja a vásárló, valamint a kiszámolt fizetendő összeg is itt jelenik meg.
* **Adatok megadása.** A kosár átnézése után a vásárló köteles megadni a személyes információit, a gyermek adatait (név, osztály/csoport) a szállítási és a fizetési módot.

## Fejlesztés tervezett menete

# Irodalomjegyzék

|  |  |
| --- | --- |
| [1] | G. Inc., „Printing,” Google Photos, [Online]. Available: https://www.google.com/photos/printing/photo-prints/. [Hozzáférés dátuma: 19 10 2023]. |
| [2] | „Digitalizáltuk az iskolai fotózást! - Ezt minden szülőnek és tanárnak látnia kell!,” 30 Március 2023. [Online]. Available: https://www.youtube.com/watch?v=\_ijP0GXR0jg. |
| [3] | „Usage statistics of server-side programming languages for websites,” w3techs, 5 10 2023. [Online]. Available: https://w3techs.com/technologies/overview/programming\_language. [Hozzáférés dátuma: 5 10 2023]. |
| [4] | Z. Powell, „Node.js vs PHP: A Head-to-Head Comparison,” Kinsta, 10 augusztus 2023. [Online]. Available: https://kinsta.com/blog/node-js-vs-php/. [Hozzáférés dátuma: 5 10 2023]. |
| [5] | „History of PHP,” php.net, [Online]. Available: https://www.php.net/manual/en/history.php.php. [Hozzáférés dátuma: 5 10 2023]. |
| [6] | J. Balfour, „The Inconsistencies of PHP,” jamiebalfour.scot, 22 4 2015. [Online]. Available: https://www.jamiebalfour.scot/articles/posts/the-inconsistencies-of-php. [Hozzáférés dátuma: 5 10 2023]. |
| [7] | „Backward incompatible changes,” php.net, [Online]. Available: https://www.php.net/manual/en/migration70.incompatible.php. [Hozzáférés dátuma: 5 10 2023]. |
| [8] | S. Ravoof, „The Definitive PHP 7.2, 7.3, 7.4, 8.0, and 8.1 Benchmarks,” Kinsta, 29 9 2023. [Online]. Available: https://kinsta.com/blog/php-benchmarks/. [Hozzáférés dátuma: 5 10 2023]. |
| [9] | R. G. Kedar Kanekar, „PHP vs ASP.NET: A Detailed Comparison,” Turing, [Online]. Available: https://www.turing.com/blog/php-vs-asp-net/. [Hozzáférés dátuma: 5 10 2023]. |
| [10] | A. Ivey, „COINTELEGRAPH,” 24 7 2023. [Online]. Available: https://cointelegraph.com/news/history-of-python-programming-language. [Hozzáférés dátuma: 19 10 2023]. |
| [11] | Z. A. Shaw, Learn Python the Hard Way, Addison-Wesley, 2014. |
| [12] | A. Bahgat, „Kinsta,” 13 9 2023. [Online]. Available: https://kinsta.com/blog/flask-vs-django/. [Hozzáférés dátuma: 19 10 2023]. |
| [13] | A. Jain, „A Brief History of .NET (dotnet),” Medium, 28 8 2023. [Online]. Available: https://medium.com/calvin-codes/a-brief-history-of-net-ec4c14adf441. [Hozzáférés dátuma: 19 10 2023]. |
| [14] | D. J. Garcia, „The History of ASP.NET – Part I,” dotnetcurry.com, 26 4 2019. [Online]. Available: https://www.dotnetcurry.com/aspnet/1492/aspnet-history-part-1. [Hozzáférés dátuma: 19 10 2023]. |
| [15] | L. S. Vailshery, „Most used programming languages among developers worldwide as of 2023,” statista, 19 7 2023. [Online]. Available: https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/. [Hozzáférés dátuma: 24 10 2023]. |
| [16] | F. Hámori, „The History of React.js on a Timeline,” RisingStack, 31 5 2022. [Online]. Available: https://blog.risingstack.com/the-history-of-react-js-on-a-timeline/. [Hozzáférés dátuma: 24 10 2023]. |
| [17] | M. Prasad, „Vue.js history,” Medium, 10 8 2023. [Online]. Available: https://madushaprasad21.medium.com/vue-js-history-1a6b8567198f. [Hozzáférés dátuma: 1 11 2023]. |
| [18] | „Virtual DOM and Internals,” React, [Online]. Available: https://legacy.reactjs.org/docs/faq-internals.html. [Hozzáférés dátuma: 24 10 2023]. |
| [19] | „Stack Overflow Trends,” Stack Overflow, [Online]. Available: https://insights.stackoverflow.com/trends?tags=reactjs%2Cvue.js%2Cangular%2Csvelte%2Cangularjs%2Cvuejs3. [Hozzáférés dátuma: 24 10 2023]. |
| [20] | „Introducing JSX,” React, [Online]. Available: https://legacy.reactjs.org/docs/introducing-jsx.html. [Hozzáférés dátuma: 24 10 2023]. |
| [21] | „Component,” React, [Online]. Available: https://react.dev/reference/react/Component. [Hozzáférés dátuma: 1 11 2023]. |

[2.1. Táblázat: Hasonló rendszerek összehasonlítása 6](#_Toc149759825)

[3.1. Táblázat Back-end környezetek és keretrendszerek összehasonlítása 11](#_Toc149759826)

[3.2. Táblázat: Frontendes keretrendszerek összehasonlítása 14](#_Toc149759827)

Ábrajegyzék

[2.2.1. ábra: Képernyőkép a Pixieset weboldalról (Forrás: https://bondi.pixieset.com/wassermanfamily/) 4](file:///C:\Users\torok\source\repos\szakdolgozat\szakdolgozat\szakdolgozat.docx#_Toc149759850)

[3.2.1. ábra: A Stack Overflow weboldalon feltett kérdések száma frontend technológiák szerint [18] 13](file:///C:\Users\torok\source\repos\szakdolgozat\szakdolgozat\szakdolgozat.docx#_Toc149759851)

[4.2.1. ábra: React osztálykomponensek (forrás: https://medium.com/@tgholami/react-function-components-lifecycle-8a6ede4b8c2e) 15](file:///C:\Users\torok\source\repos\szakdolgozat\szakdolgozat\szakdolgozat.docx#_Toc149759852)

[4.2.2. ábra: Példa a useState hook használatára (forrás: saját ábra) 16](file:///C:\Users\torok\source\repos\szakdolgozat\szakdolgozat\szakdolgozat.docx#_Toc149759853)