

Pannon Egyetem
Műszaki Informatikai Kar
Rendszer- és Számítástudományi Tanszék
Üzem-mérnök-informatikus BSc

SZAK-/DIPLOMADOLGOZAT

Logopédia időpont foglaló webalkalmazás

Csáktornyai Ádám József

Témavezető: Baumgartner János

2023



PANNON EGYETEM

MŰSZAKI INFORMATIKAI KAR

Üzemméternök-informatikus BProf szak

Veszprém, 2024. március 30.

SZAKDOLGOZAT TÉMAKIÍRÁS

Csáktornyai Ádám József

Üzemméternök-informatikus BProf szakos hallgató részére

Logopédiai időpont- és eszközfoglaló webalkalmazás

Témavezető: Dr. Baumgartner János, Egyetemi adjunktus

A feladat leírása:

Cél olyan webes alkalmazás fejlesztése, amely informatikai támogatást nyújt egy logopédiai folyamatokkal foglalkozó szolgáltatás hatékonyságának növelésére. Az applikáció két fő része az okos, személyre szabott időpontfoglalás és más pedagógusok vagy szülők számára fejlesztő eszközök kölcsönzését megvalósító funkcionálisok. További célkitűzés, hogy a web alapú fejlesztés mobilapplikációként is működjön.

Feladatkiírás:

- Dolgozza fel a témával kapcsolatos eddigi hazai és külföldi irodalmat!
- Valósítsa meg a felhasználói autentikációt, az időpontfoglalást és további statikus funkciókat, mint az eszközkölcsönzést, admin számára a foglalkozások, időpontok és kölcsönözhető eszközök kezelését valamint az időpont foglaláshoz szükséges paraméterek beállítását!
- Tervezze meg a szoftverhez szükséges adatstruktúrát, algoritmusokat!
- Tervezze meg a szoftverhez szükséges adatrepresentációt!
- A megvalósításhoz használjon valamilyen szkript nyelvet!
- Az elkészített alkalmazásnak meg kell felelnie a PWA által támasztott követelményeknek!

Dr. Vassányi István
egyetemi docens
szakfelelős

Dr. Baumgartner János
egyetemi adjunktus
témavezető

Hallgatói nyilatkozat

Alulírott Csáktornyai Ádám József hallgató kijelentem, hogy a dolgozatot a Pannon Egyetem Rendszer- és Számítástechnikai Tanszékén készítettem az Üzemmmérnök informatikus BSc végzettség megszerzése érdekében.

Kijelentem, hogy a dolgozatban lévő érdemi rész saját munkám eredménye, az érdemi részen kívül csak a hivatkozott forrásokat (szakirodalom, eszközök stb.) használtam fel.

Tudomásul veszem, hogy a dolgozatban foglalt eredményeket a Pannon Egyetem, valamint a feladatot kiíró szervezeti egység saját céljaira szabadon felhasználhatja.

Dátum: Veszprém, [év hónap nap]

.....
<<hallgató neve>>

Témavezetői nyilatkozat

Alulírott <<témavezető neve>> témavezető kijelentem, hogy a dolgozatot <<hallgató neve>> a Pannon Egyetem <<tanszék neve>>én készítette <<végzettség>> végzettség megszerzése érdekében.

Kijelentem, hogy a dolgozat védelemre bocsátását engedélyezem.

Dátum: Veszprém, [év hónap nap]

.....
<<témavezető neve>>

Köszönetnyilvánítás

A szakdolgozatom elkészítése során számos személy támogatta és segítette munkámat. Szeretném ezúton megköszönni mindazoknak a segítséget, akik részt vettek ebben a folyamatban, és hozzájárultak ahhoz, hogy a dolgozatom a legjobb minőségű legyen.

Először is, köszönetemet szeretném kifejezni Dr. Baumgartner Jánosnak, aki a dolgozatom témavezetője volt. Valószínűleg nem értem volna el ilyen sikeres eredményeket és nem jutottam volna el idáig a szakdolgozatommal a témavezetőm segítsége és támogatása nélkül. Az általa adott útmutatás, az iránymutatások és a visszajelzések nagyban hozzájárultak ahhoz, hogy a dolgozatom a lehető legjobb formáját ölthesse. Szeretnék köszönetet mondani családomnak és barátaimnak is, akik támogattak és motiváltak a szakdolgozat írása során. Az ő támogatásuk és bátorításuk nélkül nem tudtam volna ilyen hatékonyan és eredményesen dolgozni. Köszönöm, hogy mindig mellettem álltatok, és segítettetek átlendülni az akadályokon. Még egyszer köszönöm mindenkinek a segítséget, amit kaptam. Nagyon büszke vagyok arra, amit elértünk, és biztos vagyok benne, hogy ezen segítségnek köszönhetően sokkal könnyebb volt átvészelni ezt a nehéz időszakot.

Tartalmi összefoglaló

Tartalmi összefoglaló magyarul. Az összefoglalónak tartalmaznia kell (rövid, velős és összefüggő megfogalmazásban) a következőket:

- téma megnevezése,
- megoldott feladat megfogalmazása,
- megoldási mód,
- elért eredmények,
- kulcsszavak (4-6 darab)
- terjedelme nem lehet több 1 A4-es oldalnál.

Az összefoglalót magyar és angol nyelven kell készíteni. Sorrendben a dolgozat nyelvével megegyező kerül előrébb. A cím Title stílusú, formázása: Times New Roman, nagybetű, 14 pt, félkövér, középre igazított; az összefoglaló Normál stílusú, formázása: Times New Roman, 12 pt, sorkizárt, 1.5-ös sortávolság.

Kulcsszavak: [4-6 kulcsszó felsorolása, vesszővel elválasztva]

Abstract

Tartalomjegyzék

Jelölésjegyzék	9
1. Irodalmi áttekintés	10
1.1. Logopédiai folyamatok	10
1.2. IT eszközök és szoftverek fejlődése	11
1.2.1. Operációs rendszerek	11
1.2.2. Webböngészők	11
1.3. Mobil alkalmazások fejlődése	12
1.4. PWA technológia	13
1.5. Mobil operációs rendszerek és sajátosságaik	14
1.6. Logopédia és gyógypedagógia területén használt alkalmazások felmérése	14
2. Feladat és követelmények	15
2.1. Funkcionális követelmények	16
2.1.1. Regisztráció és bejelentkezés	16
2.1.2. Időpontfoglalás	16
2.1.3. Eszköz kölcsönzés	17
2.1.4. Adminisztrátor eszközök	17
3. Felhasznált technológiák	18
3.1. Git, GitHub és GitHub Desktop	18
3.2. Visual Studio Code	19
3.3. Futtatókörnyezet és csomagkezelés	21
3.3.1. Node.js és npm	22
3.3.2. Bun	23
3.4. Programnyelvek	24
3.4.1. JavaScript	24
3.4.2. TypeScript	25
3.4.3. JSX	26
3.5. React és Redux	27
3.6. Firebase	28
3.6.1. Authentication	29
3.6.2. Hosting	29

3.6.3.	Firestore Database	30
3.6.4.	Realtime Database	31
3.6.5.	Storage	31
3.7.	PWA	32
3.8.	Google Chrome, mint fejlesztői eszköz	33
3.8.1.	Lighthouse	33
3.8.2.	React fejlesztői bővítmények.....	34
3.9.	PWA kompatibilis operációs rendszerek.....	35
3.9.1.	Android.....	35
3.9.2.	iOS	36
3.9.3.	Windows.....	36
4.	Fejlesztői dokumentáció.....	37
4.1.	Architektúra és technológiák	37
4.1.1.	Frontend.....	37
4.1.2.	Backend	40
4.1.3.	API.....	41
4.2.	Fejlesztői környezet beállítása és projekt konfigurálása	43
4.2.1.	Projekt beállítása és build-elés	43
4.3.	Adatreprezentáció	44
4.4.	Időpont foglalás működése	45
4.5.	Tárgy kölcsönzés működése	46
4.6.	Tesztelés és karbantartás.....	47
4.7.	Továbbfejlesztési lehetőségek	48
5.	Felhasználói dokumentáció	49
5.1.	Webalkalmazás elérése és telepítés	49
5.2.	Felhasználói felület és bejelentkezés	50
5.3.	Időpont foglalás	53
5.4.	Tárgy kölcsönzés	54
5.5.	Admin felület.....	54
6.	Irodalomjegyzék	55
	Mellékletek	56

Jelölésjegyzék

- AI: Artificial Intelligence (Mesterséges Intelligencia)
- GPU: Graphical Processing Unit (Grafikus Processzor / Grafikus Feldolgozó Egység)
- API: Application Programming Interface (Alkalmazásprogramozási Felület)
- CPU: Central Processing Unit (Központi Feldolgozó Egység / Processzor)
- GUI: Graphical User Interface (Grafikus Felhasználói Felület)
- HCI: Human Computer Interaction (Ember-gép kapcsolat)
- CIS: Cognitive Information System (Kognitív információs rendszer)

1. Irodalmi áttekintés

1.1. Logopédiai folyamatok

A logopédus a hang, a beszéd és a nyelvi teljesítmények fogyatékoságával, zavarával, hibájával és hátrányával küzdő gyermekek és felnőttek pedagógiai megalapozottságú gyógyító-nevelését végző szakember. A logopédus munkájában komplex ellátásra törekszik, tehát feladata a beszéd-rendellenességek kialakulásának megelőzése, felismerése és rendszeres szűrése, teljes körű vizsgálata, a beszéd és nyelvi fejlődési elmaradással küzdő gyermekek egyéni vagy csoportos fejlesztése, a beszédzavarok kezelése/terápiája, a beszédhibák korrekciója, az olvasás- és írászavarok redukciója, továbbá a logopédiai tanácsadás, kísérés és utógondozás. [1] [2]

Egy magán logopédusnak sokféle feladata van. Néhány közülük:

- Első feladata, hogy felmérje a páciens állapotát. Ehhez különböző teszteket és vizsgálatokat végezhet, például beszédvizsgálatot, nyelvvizsgálatot, vagy nyelési vizsgálatot. A vizsgálatok eredményeit felhasználva kidolgozza a beteg egyéni kezelési tervét. A kezelési terv célja, hogy a beteg a lehető leghamarabb megszabaduljon a beszéd-, nyelv-, kommunikációs, vagy nyelési problémáitól. A magán logopédus a kezelési terv szerint végzi a foglalkozásokat, amelyek során a beteg a szükséges készségeket és tudást sajátítja el. A foglalkozások során a logopédus motiválja és támogatja a beteget, hogy sikeresen teljesítse a feladatokat.
- A magán logopédusnak fontos feladata, hogy időpontokat foglaljon és lemondjon a betegek számára. Emellett egyéb adminisztratív feladatokat is el kell végeznie.
- A magán logopédusnak marketing és reklám tevékenységet is kell folytatnia, hogy minél több pácienshez eljusson.
- A fejlesztő játékok és eszközök használata a beszéd-, nyelv-, kommunikációs, vagy nyelési problémák kezelésének hatékony eszköze lehet. A megfelelően kiválasztott játékok és eszközök segíthetnek a gyerekeknek a beszédértés, a beszédprodukció, a nyelvtan, a kommunikáció, vagy a nyelés fejlesztésében.

A szakdolgozatom ezen feladatok egy részében nyújt segítséget a logopédusoknak és gyógypedagógusoknak.

1.2. IT eszközök és szoftverek fejlődése

Az informatikai hardverek folyamatosan fejlődnek. A processzorok sebessége növekszik, a memóriák kapacitása bővül, és az adattároló eszközök egyre nagyobb tárhellyel rendelkeznek. Ez lehetővé teszi a komplexebb alkalmazások futtatását és a webalkalmazások hatékonyabb működését.

A mobil eszközök (például okostelefonok és táblagépek) is jelentős fejlődésen mentek keresztül. Ezek a készülékek ma már teljes értékű platformként szolgálnak, amelyeken futtathatunk nem csak natív, de webalkalmazásokat is.

1.2.1. Operációs rendszerek

Az operációs rendszerek kulcsfontosságú szerepet játszanak az alkalmazások és webalkalmazások támogatásában. Az operációs rendszerek fejlődése lehetővé teszi a jobb teljesítményt, a magasabb biztonságot és az egyszerűbb felhasználói élményt.

A legnépszerűbb operációs rendszerekben már beépített funkció a webalkalmazások alkalmazásként futtatása:

- Android-on és iOS-en a webalkalmazások kikerülhetnek főképernyőre, lehetnek saját beállításai, adataik és az appok használhatnak eszközben lévő szenzorokat a natívabb élményhez.
- Windows alatt a webalkalmazások saját, külön ablakban futhatnak és a Start menüből is indíthatók a számítógépre / laptopra telepített programokhoz hasonlóan. Szintén saját ikonnal és beállításokkal rendelkeznek.

1.2.2. Webböngészők

A webböngészők fejlődése az informatika történetének egyik legfontosabb és legdinamikusabb területe. A webböngészők olyan szoftverek, amelyek lehetővé teszik a felhasználók számára, hogy weblapokat böngésszenek, tartalmakat megtekintsenek és online szolgáltatásokat használjanak. A fejlődésük során számos technológiai változást és innovációt hoztak magukkal.

- Az első webböngésző, a WorldWideWeb, 1990-ben készült Tim Berners-Lee által. Ezt követte a Mosaic, amely 1993-ban jelent meg és az első grafikus böngésző volt.

- A Netscape Navigator a 90-es évek közepén vált népszerűvé, és hozzájárult a web elterjedéséhez.
- A 90-es évek végén és a 2000-es évek elején a böngészőháborúk zajlottak. Az Internet Explorer és a Netscape Navigator versengtek a piaci részesedésért.
- Az Internet Explorer hosszú ideig dominált, de később a Mozilla Firefox, majd a Google Chrome is erős versenyt jelentett. [3]
- A mai modern böngészők, mint például a Google Chrome, a Mozilla Firefox, az Apple Safari és az Microsoft Edge, sokkal gyorsabbak, biztonságosabbak és kompatibilisebbek.
- A HTML5 és CSS3 technológiák támogatása lehetővé teszi az interaktívabb és multimédiás tartalmak megjelenítését. [4]

A mai modern böngészők már mind támogatják a progresszív webalkalmazásokat és lehetővé teszik a támogatott operációs rendszereiken ezeknek natív szerű futtatását.

1.3. Mobil alkalmazások fejlődése

A mobil eszközök és az azokon futó alkalmazások terén az elmúlt években látványos fejlődés tapasztalható. Az okostelefonok és tabletek terjedése és elterjedése lehetővé tette az emberek számára, hogy bármikor és bárhol hozzáférjenek az internethez és az online szolgáltatásokhoz. Az alkalmazásfejlesztés terén is számos változás történt. Korábban az alkalmazásokat külön-külön kellett fejleszteni az Android és az iOS rendszerekhez, ami nagyobb költségekkel és idővel járt. Azonban az utóbbi években az úgynevezett cross-platform megoldások lehetővé teszik az alkalmazások egyszerre történő fejlesztését mindkét platformra, csökkentve a fejlesztési költségeket és időtartamot.

Ezzel párhuzamosan, a webtechnológiák terén is számos fejlesztés történt. A progresszív webalkalmazások (PWA-k) lehetővé teszik, hogy egy weboldal alkalmazás-szerűen működjön, offline üzemmódot és értesítéseket biztosítva a felhasználók számára. Az ilyen típusú alkalmazások előnye, hogy egyetlen kódbázisból indulnak, és azonnal futtathatóak a böngészőben, azonban számos olyan funkció is rendelkezésre áll, amelyeket korábban csak a natív alkalmazások biztosítottak.

Ezen trendek figyelembevételével a logopédiai folyamatok támogatását megvalósító alkalmazásom is PWA-ként fejlesztem, így lehetővé téve a felhasználók

számára, hogy akár telefonjuk böngészőjéből is hozzáférjenek az alkalmazás szolgáltatásaihoz, illetve admin felülethez.

1.4. PWA technológia

Az utóbbi években egyre népszerűbbé váltak a PWA-k (Progressive Web Applications), amelyek egyesítik a webalkalmazások és a natív alkalmazások előnyeit. A PWA-k az internet böngészőjében futnak, de telepíthetők és olyan funkcionalitást kínálnak, amely hasonló a natív alkalmazásokhoz. A PWA-k az utóbbi időben egyre elterjedtebbek lettek a vállalatok és az üzleti felhasználók körében, mert könnyen fejleszthetők és frissíthetők, valamint a felhasználók is hasonló élményt tapasztalhatnak általuk, mint a natív alkalmazások esetén.

A PWA-knak számos előnyük van a hagyományos webalkalmazásokkal és a natív alkalmazásokkal szemben. Egyik legnagyobb előnye, hogy a felhasználók a böngészőjükből telepíthetik a PWA-kat, így nem szükséges az alkalmazások letöltése és telepítése a készülékekre, ami időt takarít meg és nem foglal helyet a felhasználói eszközökön. A PWA-k gyorsak és reszponzívok, és a felhasználók offline módban is használhatják őket, amely azok számára előnyös, akik korlátozott hozzáféréssel rendelkeznek az internethez. Bár a webalkalmazásom több funkciójához is szükséges internetkapcsolat, vannak aloldalak és komponensek, amelyek kihasználhatják az offline használat előnyeit. A PWA-k használata nem igényel különösebb technikai tudást, és a fejlesztőknek nem szükséges több platformra külön-külön alkalmazásokat fejleszteniük, ami időt takarít meg és egyszerűsíti a fejlesztési folyamatot.

Az előnyök mellett azonban vannak korlátjai is a PWA-knak, például korlátozottabb hozzáféréssel rendelkeznek a készülék hardveres erőforrásaihoz, mint a natív alkalmazások.

Az alkalmazás elkészítése mellett az egyik legfőbb motiváció az volt, hogy a logopédiai területen tevékenykedő szakemberek munkáját szeretném támogatni egy olyan alkalmazással, amely a legmodernebb technológiákat használja. A PWA technológia választása pedig nem véletlen, hiszen számos előnye miatt könnyen hozzáférhető és használható. Egyrészt minden platformon elérhető, így nem szükséges külön natív alkalmazásokat fejleszteni különböző platformokra. Emellett a PWA alkalmazások használata szinte azonnal elkezdhető, nem igényel telepítést és használatuk során a felhasználói élmény is közelítheti a natív alkalmazások szintjét. Ezek mind

nagyon fontos szempontok, hisz az embereknek, ha segítségre van szükségük, nagy valószínűséggel nem a telefonjuk alkalmazás áruházába fordulnak először.

1.5. Mobil operációs rendszerek és sajátosságaik

Az okostelefonok már szinte minden embernél megtalálhatóak. A két leghasználtabb rendszer ezeken az eszközökön pedig az Android és az iOS. Bár sok hasonlóság van a kettő között, egy natív alkalmazás fejlesztése különböző módszerekkel, eszközökkel és lépésekkel történik.

Android esetén bármilyen operációs rendszeren lehet natív alkalmazást készíteni (akár magán Android-on is) Java vagy Kotlin (vagy alternatív környezetekben C#, JavaScript, TypeScript vagy C++) nyelven. iOS-nél pedig szükség van egy Macbook-ra, hogy build-elni tudjuk az alkalmazást, mindegy milyen környezetben vagy nyelven fejlesztettük.

Ezek a különbségek megnehezítik az egyéni fejlesztők és cégek dolgát. Bár már vannak technológiák arra, hogy egy kóddal 2 platformra egyaránt lehessen build-et készíteni, például a React Native, Flutter, a nemrég került stabil állapotba Kotlin Multiplatform Mobile és sok más, de még így is rengeteg rendszerbéli különbségre kell odafigyelni a fejlesztés során.

A fentebb említett PWA technológiánál viszont nem kell annyira aggódni, hogy a kód máshogy működik a különböző operációs rendszereken vagy máshogy néz ki különböző kijelzőkön. De ez nem jelenti azt, hogy nincsenek különbségek.

Egy progresszív webalkalmazás működéséhez a felhasználónak szüksége van egy böngészőre. Minden rendszeren létezik alapértelmezett böngésző, de ezek nem mindig ugyanazok. Például Android (és Windows) operációs rendszereken a chromium motor jeleníti meg a PWA-kat, de iOS (és MacOS) rendszerek esetén a Safari webböngésző látja el ezt a feladatot. Ha párosítjuk a PWA-t egy modern keretrendszerrel, ami támogatja ezeket a böngészőket, akkor szinte semmilyen különbség nincs az Android-on és iOS-en (és minden más) futó progresszív webalkalmazás működésében és megjelenítésében.

1.6. Logopédia és gyógypedagógia területén használt alkalmazások felmérése

Bár az informatika gyorsan fejlődik, a magyar oktatásban alig látható ez a fejlődés az általános- és középiskolák körül. Viszont vannak törekvések a haladásra. Néhány helyen

a logopédiai és gyógypedagógiai foglalkozásokon tableteket használnak oktató, fejlesztő játékokkal.

A foglalkozásokon kívül, reklámozás terén is lehet látni igyekezetet az informatikai lehetőségek alkalmazására. Több magán tanár és logopédus egyszerűbb weboldalt hoz létre a szolgáltatásainak reklámozására. Ezeken a weboldalakon általában szerepel bemutatkozás, elérhetőségek és ár lista. Ritkább esetekben letölthető feladatokat is észrevettem. [5] Innen jött a fejlesztő játék és tárgy kölcsönzés ötlete. Nem csak tartalomban, de technikailag is egyszerűnek mondhatók az ilyen weboldalak. Kutatásom során úgy láttam, hogy gyakran online, vizuális weboldal készítő eszközzel hozták létre ezeket az oldalakat.

Ezek a szempontokon kívül nem igazán látható más alkalmazása az informatikának, tehát a szakdolgozatom egy informatika szempontból még szinte új területén fog működni.

2. Feladat és követelmények

A szakdolgozat célja egy Logopédiai időpontfoglaló és eszköz kölcsönző webalkalmazás fejlesztése, amely kifejezetten logopédusok és pácienseik igényeire szabott. Az általános szándék az, hogy lehetővé tegye a logopédusok számára az online időpontfoglalást, és egyszerűsítse a gyógypedagógiai eszközök kölcsönzését más pedagógusok és logopédusok számára.

A szakdolgozat tervezett feladatkörei közé tartozik egy alkalmazkodó időpontfoglalási rendszer kialakítása, az eszközkölcsönzési rendszer implementálása, valamint a felhasználói fiókok kezelése. Az alkalmazásnak kiemelt szempont, hogy felhasználóbarát és intuitív felülettel rendelkezzen, lehetővé téve mind a logopédusok, mind pedig a páciensek számára egyszerű használatot.

A tervezett alkalmazásnak számos fontos tulajdonsággal kell rendelkeznie. Az alkalmazkodó időpontfoglalási rendszernek lehetővé kell tennie a logopédusok számára, hogy rugalmasan és hatékonyan menedzseljék a rendelési időpontjaikat. Ezenkívül az eszközkölcsönzési rendszernek lehetővé kell tennie más pedagógusoknak, logopédusoknak stb., hogy könnyedén és gyorsan hozzáférjenek és kölcsönözzenek gyógypedagógiai eszközöket.

A felhasználói fiókok kezelése során kiemelt figyelmet fordítunk a biztonságra és az adatvédelemre. Az alkalmazásnak stabilnak, biztonságosnak, modernnek és könnyen karbantarthatónak kell lennie. Ezeket a célokat a legújabb technológiák alkalmazásával, alapos kutatómunkával és olvasható, rendezett kód írásával igyekszem elérni.

A tervezett alkalmazás fejlesztése során korszerű technológiákat alkalmazok, figyelembe véve a szoftvertervezés legújabb trendjeit. Az alkalmazás kliensoldalán a progresszív webalkalmazás (PWA) elveit alkalmazom a felhasználói élmény javítása érdekében. Az alkalmazás szerveroldalán pedig egy megbízható és hatékony webalkalmazás keretrendszerrel valósítom meg a funkcionális követelményeket.

2.1. Funkcionális követelmények

A szakdolgozat tervezésénél a webalkalmazás főbb funkcióit a témavezetőmmel alaposan átbeszéltük, hogy hogyan kell működniük és milyen funkcionális követelményeknek kell teljesülnie.

2.1.1. Regisztráció és bejelentkezés

- A felhasználóknak legyen lehetősége egyszerűen fiókot létrehozni egy email cím és jelszó használatával.
- Legyen opció Google fiókkal való belépésre, amivel akár a regisztráció folyamata is kihagyható.
- A regisztrálás és bejelentkezés felülete legyen könnyen elérhető.
- A webalkalmazás emlékezzen a felhasználókra és ne kelljen mindig újra bejelentkezni ugyanazon az eszközön, még akkor sem, ha az alkalmazást bezárta a felhasználó vagy újraindította az eszközt.
- A felhasználóknak legyen egyszerű módja a kijelentkezésre.

2.1.2. Időpontfoglalás

- A felhasználóknak legyen lehetősége időpontot foglalni a logopédusnál.
- Az időpontfoglalás legyen egyszerű és intuitív. A háttérben a rendszer okosan kezelje a választható időpontokat, hogy ne legyen lehetőség két páciensnek ugyanarra vagy problémás időben jelentkezni. Figyelembe vett szempontok:

- Ne legyen ütközés más időpontokkal vagy a logopédus szabadnapjaival.
- Az időpontok legyenek a logopédus által beállított munkaidőben.
- Legyen egy érték, amely az időpontok közötti minimális szünetet szabályozza. Ez az érték változzon a logopédus beállításaitól és az adott napon már lefoglalt időpontoktól függően.
- A felhasználók tudjanak időpontot lemondani.
- A logopédus tudjon időpontokat megtekinteni, módosítani és törölni.

2.1.3. Eszköz kölcsönzés

- A felhasználói oldalon legyen egy gomb, amivel böngészhetők a kölcsönözhető fejlesztő játékok és eszközök.
- A felhasználók lássák, hogy egy eszköz kölcsönözhető-e vagy sem.
- A felhasználóknak legyen lehetősége tárgyakat lefoglalni, amit az adminisztrátor lát az ő oldalán és látja, hogy melyik felhasználó foglalta le az eszközt.
- Az adminnak legyen lehetősége szerkeszteni az eszközöket és azok állapotát.
- A kölcsönözhető eszközöknek legyen képe, amit az admin URL és saját eszközén lévő fájl feltöltésével is beállíthat. Ez a kép jelenjen meg a felhasználók számára is böngészésnél és az admin számára is szerkesztés közben.

2.1.4. Adminisztrátor eszközök

- A logopédusnak legyen egy külön felülete, ahol saját beállításai vannak és kezelheti a webalkalmazásban lévő tartalmakat.
- A logopédusnak legyen lehetősége beállítani a munkaidőt, szabadnapokat és a minimális szünetet az időpontok között.
- A logopédusnak legyen lehetősége hozzáadni és törölni eszközöket és szerkeszteni a létező kölcsönözhető eszközök tulajdonságait.
- A logopédusnak legyen lehetősége az időpontokat megtekinteni, módosítani és törölni.

3. Felhasznált technológiák

Az informatika egyre csak fejlődik komplexebb és hatékonyabb technológiákkal. Ez nagyrészt a programozók közösségének és együttműködésének köszönhető. Rengeteg projekt létezik, amelynek az a célja, hogy más programozók munkáját könnyítsék meg. Ezért is kedvelem az informatika szakmát nagyon, mert a saját munkaeszközeinket készíthetjük, alakíthatjuk a feladatainkhoz és mi, mint fejlesztők támaszkodhatunk egymásra.

3.1. Git, GitHub és GitHub Desktop

A Git a legelterjedtebb verziókezelő rendszer a szakmában. Sokan nélkül nem is kezdenek projektet. Magam is hasonlóan állok neki egy munkának, hisz a Git verzió követő és kezelő funkciói elengedhetetlenek egy nagyobb szoftver projekt során.

A Git-tel már középiskola óta vannak tapasztalataim és többször is segítettek elhárítani nagyobb problémákat bonyolultabb projekteknél. Mivel a Git lehetőséget ad, hogy bármilyen korábbi állapotát visszaállítsuk egy projektnek, így a fájlok sérülésének vagy elvesztésének kockázata megszűnik. Illetve a Git mellett határozottabban lehet egy projektben kísérletezni, új dolgokat kipróbálni és egy projekt fontosabb részeit lecserélni. Például ebben a projektben is út közben programnyelv váltás történt, amit a Git-nek köszönhetően teljes biztonsággal, új hibák felmerülésének félelme nélkül hajthattam végre.

A biztonság érdekében a munkáim általában több helyen tárolom, biztonsági mentéseket készítek. Többek között ebben is segít a GitHub, amely egy Git alapú internetes szolgáltatás. Itt több, hasznos funkció várja a fejlesztőket a kód felhőben tárolásán kívül, mint például host-olás, automatikus build-elés, online fejlesztés, együttműködés stb. Már középiskola eleje óta használom személyes és munkahelyi projekteknél és számtalanszor igénybe vettem az említett funkciókat.

A Microsoft fejlesztésű weboldal minden fejlesztő számára ismert és szinte mindenki használja. A jelenlegi cégemnél is itt dolgozunk együtt, tartjuk számon a munkát, kezeljük a kiadásokat stb.

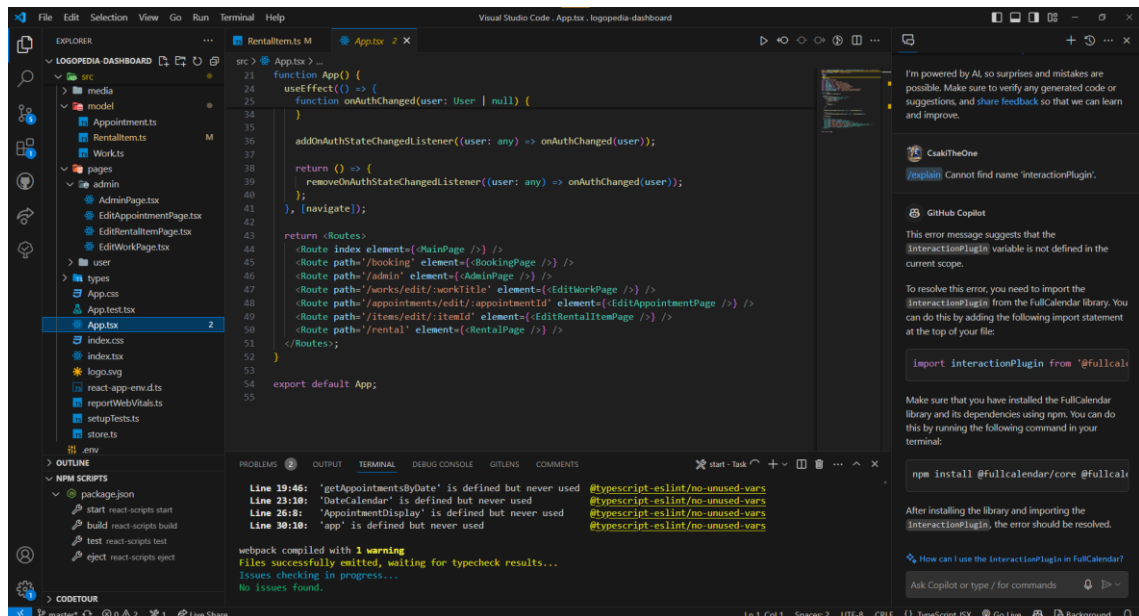
Szoftverfejlesztésen kívül is hasznos eszköznek bizonyul bármilyen más, digitálisan tárolható projektekhez és jegyzetekhez. Egyetemi éveim alatt itt tároltam az

órai jegyzeteim, csoportmunkánál itt dolgoztunk együtt és az Android projektjeim adatvédelmi irányelvei is itt vannak. A GitHub egy határozottan sokoldalú szolgáltatás és ezt tanulmányim és munkáim során mindig igyekeztem kihasználni.

Hogy gördülékenyen menjen a munka és egyszerűen végezhessem a verziók kezelését, a GitHub Desktop programot használom. Ez a GitHub hivatalos asztali kliense. Bár a választott fejlesztői környezetem beépítetten támogatja a GitHub-ot és teljesen beépül a programba, mégis ezt a külön programot használom az egyszerűsége és felhasználóbarátsága miatt. Ezen kívül gyakran váltok a munkahelyi és személyes fiókom között. A GitHub Desktop-pal pár kattintásba telik ez a művelet, így könnyen visszatérhetek a szakdolgozathoz munka után.

3.2. Visual Studio Code

A Visual Studio Code (VSCode) az egyik legnépszerűbb kódszerkesztő program a szoftverfejlesztők körében. Középiskola óta használom ezt is különböző személyes, munkahelyi és iskolai projekthez. Ez nem véletlen, hisz a bővítményeknek köszönhető testreszabhatósága miatt számos feladtnál jó választásnak bizonyul. Legyen az weboldal, Python script, React Native alkalmazás, iskolai jegyzet vagy mod egy kedvenc játékhoz.



Minden projekt kezdetén, amelynél a Visual Studio Code programot használom, a tervezett projekthez illő bővítményeket keresek, hogy a fejlesztésnél gyorsabb és hatékonyabb legyek. A szakdolgozatomnál sem tettem másként, viszont a

munkahelyemen lévő feladataim miatt már elő volt készítve az ehhez a webalkalmazáshoz hasznosnak talált néhány kiegészítő.

- VSCode hasznos beépített modulok
 - Emmet: gyors és egyszerű HTML, JSX és TSX szerkesztés
 - JavaScript és TypeScript nyelv szervert: kód kiegészítés, szintaxis színezés, import javaslatok, inline dokumentáció JS és TS nyelvekhez
 - NPM támogatás: NPM csomagkezelő támogatás a VSCode-ba, amely többek között egyszerűvé teszi az NPM script-ek futtatását és az ezt használó projektek hibakeresését
- fontosabb bővítmények
 - ESLint: formázás és fejlesztés idejű hibakeresés a kódban JS és TS nyelvekhez
 - PWABuilder Studio: PWA feltétel ellenőrző és kód generáló valid PWA készítéshez
 - VS Code ES7+ React/Redux/React-Native/JS snippets: kódkiegészítés, terjedelmesebb kódjavaslatok és újra felhasználható kódrészletek React alapú projektekhez
 - GitLens: inline információk git alapú projektekhez, ami megmutatja, hogy melyik sor mikor változott utoljára
 - GitHub Copilot: mesterséges intelligencia, amely szinte bármilyen projektben tud terjedelmes, pontos és projekthez illő kódjavaslatokat adni
 - GitHub Copilot Chat: az előző pontban említett mesterséges intelligenciának egy külön felületet ad ez a bővítmény, hogy a kódszerkesztőn kívül, de a projekt fájljait felhasználva is tudjon segíteni a fejlesztés során
 - Live Preview: helyi hálózaton futó szerverrel biztosít a kódszerkesztőben, hogy a projektet más eszközökön is ki lehessen próbálni fejlesztés közben; például én így teszteltem a webalkalmazást az Android-os és iOS készülékemen. A szerver minden változtatásnál frissíti a webalkalmazást a csatlakozott eszközökön én így gyorsan tesztelhettem a projekt különböző felhasználói felületeit kisebb képernyőn.

3.3. Futtatókörnyezet és csomagkezelés

A webfejlesztés és weboldalak sokat fejlődtek az évtizedek során. A kezdeti időkben a weboldalak jellemzően statikusak voltak, ami azt jelenti, hogy az oldal tartalma fix, nem változik dinamikusan a felhasználói interakciókra. Az 1990-es évek elején a HTML (Hypertext Markup Language) volt a fő alkalmazott technológia a weboldalak készítésére. Ezek a statikus weboldalak egyszerűek voltak, ám korlátozottak a felhasználói élmény terén.

A webfejlesztés dinamizálására és interaktivitás növelésére az évek során egyre inkább előtérbe került a kliensoldali (frontend) JavaScript használata. A JavaScript egy olyan programozási nyelv, amely lehetővé teszi a böngészőben történő dinamikus interakciókat. A fejlesztők így képesek voltak az oldalakat interaktívvá tenni, aszinkron kéréseket indítani, és az oldal tartalmát dinamikusan frissíteni anélkül, hogy az egész oldalt újratöltenék.

Az Ajax (Asynchronous JavaScript and XML) bevezetése tovább fokozta a dinamizmust, lehetővé téve az oldalakon belüli aszinkron adatátvitelt. Ennek eredményeként a weboldalak kezdtek elérhetőbbek lenni és a felhasználói élmény javult. Ebben az időszakban azonban a backend (szerveroldali) fejlesztés is dinamikusabbá vált, és a szerveroldalon is elkezdtek alkalmazni a dinamikus nyelveket, például a PHP-t, hogy a szerver oldali logika is interaktívabb lehessen.

Az egyik jelentős mérföldkő a webfejlesztés terén a Single Page Application (SPA) koncepció megjelenése volt. Az SPA olyan webalkalmazás, amely csak egyetlen HTML oldalt tölt be, és a tartalmat dinamikusan cseréli ki JavaScript segítségével. Ezáltal elkerülhető a teljes oldal újratöltése, ami jelentős sebességnövekedést eredményez.

Az SPA-k és a PWA-k térhódításával együtt olyan frontend keretrendszerek (frameworks) váltak népszerűvé, amelyek lehetővé teszik a fejlesztők számára, hogy hatékonyabban és strukturáltabban építsék fel az alkalmazásaikat. Ezek a keretrendszerek kínálnak különböző funkciókat és eszközöket, amelyek segítik a fejlesztőket az alkalmazások könnyebb karbantartásában, kódújrafelhasználásban és a fejlesztési folyamatok felgyorsításában.

Ebben a kontextusban olyan frontend keretrendszerek, mint a React, Vue, és Svelte, játszanak kiemelkedő szerepet. A React, a Facebook által fejlesztett és karbantartott, egy deklaratív, hatékony és rugalmas JavaScript könyvtár, amely lehetővé teszi a komponensalapú felépítést. A Vue egy másik népszerű frontend keretrendszer, amely

könnyen tanulható és integrálható meglévő projektekbe is. A Svelte pedig egy új megközelítést kínál, amely a futtatási időben történő kódgenerálást helyezi előtérbe, így a generált kód lényegesen kisebb és hatékonyabb.

Amikor ezekről a keretrendszerekről beszélünk vagy bármilyen SPA-ról, akkor fontos beszélni ezeknek a futtatásáról. Itt jön képbe a Node.js.

3.3.1. Node.js és npm

Node.js egy olyan szerveroldali JavaScript futtatókörnyezet, amelyet Ryan Dahl fejlesztett ki 2009-ben. A Node.js a V8 JavaScript motorra épül, amely eredetileg a Google Chrome böngésző része volt. Node.js lehetővé teszi a JavaScript nyelv használatát a szerveroldalon, és a szoftvertervezési paradigma aszinkron I/O (bemenet/kimenet) alapelveire épül.

Az egyik legfontosabb tulajdonsága a Node.js-nek az eseményvezérelt (event-driven) és nem blokkoló I/O. Ez azt jelenti, hogy a Node.js alkalmazások képesek aszinkron módon kezelni a bemeneteket és a kimeneteket, aminek köszönhetően hatékonyan tudnak skálázódni a nagy terhelésű alkalmazásoknál is. Például egy Node.js szerver egyszerre képes több ezer csatlakozott klienst kiszolgálni anélkül, hogy a folyamatokat blokkolná.

A Node.js alkalmazások különféle területeken használhatók, beleértve a webszervereket, API-kat, valós idejű alkalmazásokat és sok más alkalmazási területet. A NPM (Node Package Manager) segítségével könnyedén kezelhetők és telepíthetők különböző modulok, könyvtárak és keretrendszerek a Node.js alkalmazásokhoz.

Az npm egy olyan csomagkezelő, amely lehetővé teszi a fejlesztők számára, hogy könnyedén importáljanak és telepítsenek harmadik féltől származó könyvtárakat, illetve megosszák a saját kódjukat a közösséggel. Ez hatalmas előny a fejlesztőknek, mivel lehetővé teszi számukra, hogy újrahasznosítsák a kódot, és gyorsan fejlesszék az alkalmazásaikat.

A Node.js ökoszisztémája rendkívül élénk, és számos olyan keretrendszert kínál, amelyek segíthetnek a fejlesztőknek az alkalmazások gyorsabb és hatékonyabb létrehozásában. Például az Express.js egy népszerű és könnyen tanulható webalkalmazás-keretrendszer, amely segíti az útvonalak definiálását, a HTTP kérések kezelését és sok más funkciót.

A Node.js alapja az eseményvezérelt programozás és a callback függvények használata. Ezen túlmenően, a modern Node.js verziók támogatják az aszinkron kódot kezelő `async/await` mintáját is, amely lehetővé teszi a fejlesztők számára, hogy olvashatóbb és könnyebben karbantartható kódot írjanak.

Node.js rendkívül népszerűvé vált a fejlesztők körében, és a webserverek, mikroszolgáltatások, valós idejű alkalmazások területén széles körben alkalmazzák. A sokoldalúsága és a gyors fejlesztési folyamatok elősegítése miatt a Node.js az egyik kulcsfontosságú technológia a modern szerveroldali fejlesztési területeken.

3.3.2. Bun

A rengeteg technológia, csomag és keretrendszer miatt a webfejlesztés egy nagyon bonyolult témává vált az elmúlt években. Ezt próbálja orvosolni a Bun.

A Bun egy JavaScript futtatókörnyezet, amely egyben csomagoló, transzpilátor, feladatkezelő és npm-kliens is. A Bun célja, hogy egy egyszerű és gyors megoldást nyújtson a JavaScript és TypeScript alkalmazások fejlesztéséhez.

A Bun számos előnnyel rendelkezik a Node.js-hez képest:

- Gyorsabb indítási idő: A Bun futtatókörnyezet sokkal gyorsabban indul el, mint a Node.js. Ez a webalkalmazások fejlesztésekor különösen fontos, mivel a gyors indítási idő javítja az élményt a fejlesztők számára.
- Kisebb memórafogyasztás: A Bun futtatókörnyezet kevesebb memóriát fogyaszt, mint a Node.js. Ez a webalkalmazások futtatásakor különösen fontos, mivel a kisebb memórafogyasztás jobb teljesítményt eredményez.
- Egyszerűbb használat: A Bun futtatókörnyezet egyszerűbb használatú, mint a Node.js. A Bun egyetlen parancsot biztosít a webalkalmazások futtatásához, míg a Node.js-hez több parancsot kell megadni.

Viszont a Bun nem csak egy futtatókörnyezet, hanem egy komplett eszközkészlet, amely a JavaScript és TypeScript alkalmazások fejlesztéséhez szükséges összes eszközt biztosít. A Bun a következő eszközöket tartalmazza:

- Futtatókörnyezet: A Bun futtatókörnyezet egy gyors és hatékony JavaScript futtatókörnyezet, amely a webalkalmazások futtatásához szükséges.
- Csomagoló: A Bun csomagoló egy eszköz, amely a webalkalmazások fájljait egy kompakt fájlba csomagolja, amely könnyebben elérhető és szállítható.

- Transzpilátor: A Bun transzpilátor egy eszköz, amely a JavaScript kódot TypeScript kódra fordítja, ami javíthatja a teljesítményt és a biztonságot.
- Feladatkezelő: A Bun feladatkezelő egy eszköz, amely a webalkalmazások fejlesztéséhez szükséges feladatokat automatizálja.
- npm-kliens: A Bun npm-kliens egy eszköz, amely lehetővé teszi a fejlesztők számára, hogy JavaScript és TypeScript csomagokat telepítsenek és kezeljenek.

3.4. Programnyelvek

Minden szoftver készítésénél fontos kérdés a megfelelő programozási nyelv kiválasztása. a szakdolgozat témája egy progresszív webalkalmazás, ami leszűkíti a választási lehetőségek listáját, de sokat fejlődött a weboldalak készítésének módja az évtizedek során, így még mindig számos nyelv közül van lehetőségünk választani.

Az első félévben egy ismerős nyelvet szerettem volna választani, amellyel már van tapasztalatom és kényelmesen tudok dolgozni vele. Ez volt a JavaScript. Ezt követően a második félévben lecseréltem a projektben használt nyelvet TypeScript-re, amellyel úgy éreztem, hogy könnyebben tudok megvalósítani bonyolultabb funkciókat és könnyebbé teszi a hibák elkerülését a fejlesztés során.

3.4.1. JavaScript

A webalkalmazás elkészítéséhez először a JavaScript nyelvet választottam. JavaScript egy dinamikus, interpretált programnyelv, amelyet a böngészőkben történő kliensoldali fejlesztéshez terveztek, de ma már széles körben alkalmazzák mind a szerveroldali, mind a kliensoldali fejlesztés során. Netscape Communications Corporation készítette először 1995-ben, azonban ma már egy szabványosított nyelv, amelynek fejlesztését a JavaScript Standardization Organization (JSO) irányítja.

A JavaScript az ECMAScript szabvány alapján működik, és az ECMAScript definiálja a nyelv alapvető szabályait és funkcióit. A JavaScript neve gyakran összekeveredik az ECMAScripttel, de fontos megjegyezni, hogy JavaScript a böngészők környezetében is számos további API-t és lehetőséget kínál.

A JavaScript egy könnyen tanulható nyelv, ami részben annak köszönhető, hogy a szintaxisa sokban hasonlít más C-szerű nyelvekhez, például a Java-hoz vagy a C#-hoz. A nyelv dinamikus típusú, tehát a változók típusa futási időben dől el. Ez a rugalmasság

lehetővé teszi a fejlesztők számára, hogy gyorsan reagáljanak a változó követelményekre, ugyanakkor figyelmet igényel a típusbiztonság fenntartása érdekében.

JavaScript kiválóan alkalmas interaktív weboldalak készítésére. A böngészőkben történő kliensoldali használat mellett a Node.js segítségével a szerveroldalon is futtatható, amely lehetővé teszi a teljes alkalmazások kliens- és szerveroldali kódjának egységesítését.

A nyelv fejlődése során számos funkció és lehetőség került bevezetésre, például az aszinkron programozás lehetőségét biztosító Promise és a modern aszinkron szintaxist lehetővé tevő `async/await`. Az ECMAScript 6 (ES6) bevezetése jelentős lépést tett a nyelv fejlődésében, és újabb szintaktikai elemeket, osztályokat, modulokat és sok más funkciót vezetett be.

A JavaScript ma már az egyik legnépszerűbb és legelterjedtebb programozási nyelv, és a szoftvertervezés számos területén használatos, beleértve a webfejlesztést, mobilalkalmazások készítését, játékfejlesztést és sok egyéb alkalmazási területet. A folyamatos fejlesztések és a széles körű támogatás biztosítják, hogy a JavaScript továbbra is fontos szerepet játsszon a szoftvertervezés világában.

Viszont a munkám során úgy döntöttem, hogy szeretnék valami újat tanulni és kipróbálni. Így hát váltottam a TypeScript nyelvre.

3.4.2. TypeScript

A szakdolgozatom elkészítése során végül a TypeScript programnyelvet választottam, melyet gyakran típusos JavaScript-ként emlegetnek. Ez a nyelv egy fordított megközelítést alkalmaz, ahol a kódot TypeScript-ben írjuk, majd azt lefordítjuk JavaScript kóddá. Számos előnnyel rendelkezik a hagyományos JavaScript-hez képest, és ezek a javak főként a típusosságának köszönhetőek.

A tiszta JavaScript-ben a változók típusa futásidőben változhat, míg a TypeScript-ben a fejlesztő köteles minden változó típusát deklarálni, és a fejlesztői környezet figyelmeztet, ha ezt elmulasztja. Ez a szigorú típusosság rendkívül hatékonyan szűri ki a hibákat már a fejlesztés korai szakaszában, csökkentve a potenciális futási hibák kockázatát. Például a változóknak nincs null értékük, amennyiben nem adtuk meg, és minden változó csak egy típussal rendelkezhet. Emellett a TypeScript új operátorokat és műveleteket is bevezet, amelyek megkönnyítik a típusok ellenőrzését, kezelését és a null biztonságát.

Egy másik kiemelkedő tulajdonsága a TypeScript-nek, hogy bármelyik JavaScript verzióra képes fordulni. Ez azt jelenti, hogy a kódunk a böngészők, a VSCode bővítmények és a Node szerverek széles körében futtatható. Így a fejlesztőknek nem kell aggódniuk a futtató környezet támogatottsága miatt.

Ezek a pozitív tulajdonságok azonban együtt járnak néhány hátránnyal is. A típusok és a sok ellenőrzés miatt a kód hosszabb lesz. Emellett a TypeScript kódot először le kell fordítani, mielőtt azt a futtatási környezetben használnánk. Ez növeli a deploy időt, például a szakdolgozatom esetében a deploy ideje körülbelül 3 perc, ahelyett, hogy egyszerűen csak feltölteném a fájlokat a host szerverre. A fordítási lépés szükségessége tehát bizonyos mértékű komplexitást és idővesztést jelent, de ez a hátrány a fejlesztési gyorsaság és a biztonságosság szempontjából elfogadható lehetőségeket kínál.

3.4.3. JSX

Bár nem tekinteném külön programnyelvnek, érdemesnek tartom megemlíteni a JSX-et. A JSX (JavaScript XML) egy JavaScript kiterjesztés, amely lehetővé teszi XML-szerű kódszerkezet használatát a JavaScript kódokban. A React keretrendszer alapvetően JSX-t használ a komponensek létrehozásához, melyek a felhasználói felület különböző részeit reprezentálják. A JSX kódok a React elemeket és komponenseket hozzák létre, melyek renderelhetők a böngészőben vagy más környezetekben.

A JSX lehetővé teszi a dinamikus értékek és kifejezések beágyazását is a kódba. Ez azt jelenti, hogy a JavaScript változók, függvények és kifejezések közvetlenül integrálhatók a JSX kódokba, ami dinamikus és interaktív felhasználói felületek kialakítását teszi lehetővé. Például a JSX-ben könnyedén használhatunk JavaScript map függvényt egy tömbön belüli elemek renderelésére, vagy feltételes operátorokat az állapotok kezelésére.

```
<ListItem
  key={props.work.id}
  secondaryAction={
    <IconButton
      edge='end'
      onClick={() => {
        alert(props.work.description);
      }}
    >
      <HelpIcon />
    </IconButton>
  }>
  <ListItemButton
    selected={props.selected}
    onClick={props.onClick}
  >
    <ListItemText
      primary={props.work.title}
      secondary={` ${props.work.durationMinutes} > 0 &&
        props.showTime ? props.work.durationMinutes + ' perc' : ''}`
    />
  </ListItemButton>
</ListItem>
```

3.5. React és Redux

React és Redux két népszerű JavaScript könyvtár, melyeket webfejlesztéshez alkalmaznak, és szorosan kapcsolódnak egymáshoz.

A React egy deklaratív, hatékony és flexibilis JavaScript könyvtár, amelyet a felhasználói felületek fejlesztésére terveztek. A Facebook által kifejlesztett és karbantartott könyvtár, könnyen kezelhető, és lehetővé teszi a komponensalapú fejlesztést. A React alapelve az, hogy a felhasználói felületet kisebb, újrafelhasználható komponensekre bontja, amelyek egyszerűen összeállíthatók és kezelhetők. A virtuális DOM használatával a React optimalizálja a felhasználói felület frissítéseit, növelve ezzel az alkalmazás teljesítményét. A komponensalapú struktúra és az egyirányú adatáramlás segít az alkalmazások egyszerűbb tervezésében és karbantartásában.

A Redux egy állapotkezelési könyvtár, amely szorosan kapcsolódik a React-hoz, de önállóan is használható más JavaScript keretrendszerekkel. Az alkalmazás állapotának központosított tárolására és kezelésére szolgál, ami különösen hasznos nagy és bonyolult alkalmazások esetén. A Redux az egyirányú adatáramlás elvét követi, és a fejlesztőnek lehetőséget ad a következetes és előre látható állapotkezelésre. A reduktorok segítségével az alkalmazás állapotát tiszta és előre látható módon lehet módosítani. Az egyértelmű módja az adatok módosításának és a komponensek közötti adatáramlás irányításának hozzájárul a könnyen karbantartható és skálázható alkalmazások kialakításához.

A React és a Redux együttes használata kifejezetten előnyös a nagyobb és komplexebb alkalmazások esetén, ahol az állapotkezelés és a felhasználói felület kezelése központi szerepet játszik. A két technológia együttes alkalmazásával az alkalmazások hatékonyabbá, skálázhatóbbá és könnyen karbantarthatóvá válnak.

3.6. Firebase

A Firebase a Google saját backend szolgáltatása. Számos eszközt felölel a Firebase, amelyek hasznosak egy alkalmazás elkészítéséhez, teszteléséhez, felhasználóival való kapcsolatba lépéshez és még sok másához.

A legtöbb funkció egy megadott kihasználtságig teljesen ingyenes, így a Firebase szolgáltatás egy nagyon jó eszköz a kisebb fejlesztőknek a projektjeik elkészítéséhez és karbantartásához. Egyszerű választás volt nekem is backend-et választanom a szakdolgozathoz, hisz ezen a kedvező tulajdonságon kívül a tapasztalataim alapján is a Firebase mellett döntenék. Sok középiskola alatt készített projektem szervereként működik, szeretek vele dolgozni.

A Firebase több alkalmazás típusba is integrálható: Android, iOS, webalkalmazás, játék és közösség által készített eszközökkel sok más is szóba jöhet. Nekem főleg Android alkalmazásaim vannak Firebase szolgáltatással összekötve, de minden környezetben nagyon egyszerű a használata. Ezt segíti a hivatalos dokumentáció is, ahol találhatóak példakódok minden hivatalosan támogatott programozási nyelvhez (Java, Kotlin, Swift, C# és még néhány).

3.6.1. Authentication

Egy időpont foglaló alkalmazás esetén az egyik legfontosabb információ, hogy ki kéri az időpontot. A felhasználó kezelés egy összetett feladat, de a Firebase egy szolgáltatása sok részét leegyszerűsíti. A Firebase Authentication funkciója tud kezelni email jelszó kombinációs, email kódos és más szolgáltatással való regisztrálást és bejelentkezést.

A szakdolgozatomban email jelszó kombinációs és Google fiókkal való hitelesítést biztosítok a felhasználók és admin számára. Így az átlag felhasználók gyorsan be tudnak lépni az alkalmazásba és email regisztráció is van azoknak, akik nem szeretik összekötni a fiókjaikat.

3.6.2. Hosting

Az egyik fő különbség a progresszív webalkalmazások és a natív alkalmazások között, hogy míg a natív app teljesen a telefonon fut, gépi kód vagy bytecode teljesíti az utasításokat, addig a PWA esetén egy böngészőben fut az alkalmazás és a böngészőben lévő JavaScript interpreter felel a logikáért. Pontosan ezért szükségem volt egy host-ra, ahol az alkalmazást szolgáltatathatom a felhasználók és admin számára.

A Firebase szolgáltatások között host is található, ezt választottam a szakdolgozat üzemeltetéséhez. Egy bizonyos mennyiségű látogatottságig ez a szolgáltatás is teljesen ingyenes. Tökéletes a jelenlegi feladathoz, ahol nem lesz szükség több ezer felhasználó ellátására. A PWA technológia olyan szempontból is előnyös, hogy az alkalmazás részeit letölthetjük a felhasználó eszközére, ezzel is tehermentesíthetjük a választott szerverünket és ennek köszönhetően a webalkalmazás más, kisebb host-oknál is tökéletesen működne.

A projekt deploy-olása nagyon egyszerű erre a szolgáltatásra. A Firebase CLI eszközzel manuálisan, egy terminál paranccsal ki lehet adni a projektet. Viszont a szakdolgozatomban a build és deploy folyamatot automatizáltam. Mikor előkészítettem a host-ot és a projektet, a Firebase CLI program megkérdezte, hogy beállítsa-e a deploy-oló GitHub akciókat. Ezeknek köszönhetően minden alkalommal, amikor git push-olok vagy elfogadok egy pull request-et, a projektből automatikusan készül egy build és fölkerül Firebase Hosting-ra. Ez a folyamat körülbelül 3 percig tart és nagy segítség, hogy az alkalmazást több platformon is szinte azonnal tudjam tesztelni, illetve kiadás után a felmerülő hibákat gyorsan tudjam javítani.

3.6.3. Firestore Database

A Firestore Database egy valós idejű, NoSQL adatbázisszolgáltatás, amely lehetővé teszi a fejlesztők számára, hogy gyorsan és egyszerűen hozzanak létre és kezeljenek valós idejű adatokat mobil- és webalkalmazásaikban. A Firebase Database a Cloud Firestore-on alapul, amely egy teljesen menedzselt, nagy teljesítményű és biztonságos adatbázisszolgáltatás.

A Firestore Database számos előnnyel rendelkezik a hagyományos adatbázisokkal és adatbázisszolgáltatásokkal szemben:

- **Valós idejű:** A Firestore Database lehetővé teszi, hogy az alkalmazásaikba valós idejű adatokat közvetítsenek. Ez ideális például olyan alkalmazásokhoz, amelyekben a felhasználók egymással együttműködnek, például egy közösségi médiaplatformhoz vagy egy együttműködési alkalmazáshoz.
- **Könnyen használható:** A Firestore Database egyszerűen használható API-val rendelkezik, amely lehetővé teszi a fejlesztők számára, hogy gyorsan és egyszerűen hozzáférjenek és kezeljék az adataikat. Az adott projekt nyelvét használhatjuk az adatbázis kezeléséhez, így a fejlesztői környezet funkcióit itt is kihasználhatjuk a string-be írt SQL parancsokkal ellentétben.
- **Biztonságos:** A Firestore Database biztonságos adatbázisszolgáltatás, amely védi az adatokat a jogosulatlan hozzáféréstől és módosítástól. A felhasználók jogosultságait egy egyszerű, könnyen értelmezhető JSON fájl szerkesztésével tehetem meg.

A szakdolgozatomban a Firestore Database-t használtam a logopédiai időpontfoglaló és eszköz kölcsönző alkalmazásom adatainak tárolására. Az alkalmazásban a következő adatokat tárolom az adatbázisban:

- felhasználók
- időpontok
- logopédus szabadnapjai
- kölcsönözhető eszközök

Az Firestore Database gyűjteményekben és dokumentumokban tárolja a strukturált adatokat, ezért tökéletes választás volt az imént említett adatok tárolásához. Ezen adattípusokat bővebben kifejttem a fejlesztői dokumentációban. A Firebase Database használata lehetővé tette számomra, hogy egyszerűen és hatékonyan tároljam az

alkalmazásom adatait. A valós idejű adatbázis funkció lehetővé teszi, hogy a felhasználók azonnal frissítsék az időpontjaikat vagy az eszközök állapotát.

3.6.4. Realtime Database

A Firebase Realtime Database is egy valós idejű, NoSQL adatbázisszolgáltatás, amely lehetővé teszi a fejlesztők számára, hogy gyorsan és egyszerűen hozzanak létre és kezeljenek valós idejű adatokat mobil- és webalkalmazásaikban. A Firebase Realtime Database a Google Cloud Platform egyik eleme, és a Firestore elődje. Bár ez az adatbázis nem tud strukturált, egybefüggő adatokat tárolni olyan könnyen és inkább primitív adattípusok kezelésére lett kitalálva, a Realtime Database-nek is megvannak a Firestore Database-ben említett előnyei.

A fő különbség maga az adatszerkezet: Amíg a Firestore gyűjteményekben és dokumentumokban tárolja az adatokat, a Realtime Database sokkal kötetlenebb. Egy fa adatszerkezetben raktározódnak az adatok. Bármilyen típusú adatot tárolhatunk, ami egy JSON fájlban is tárolható és azon felül dátumokat, időpontokat is. Bár e különbség miatt itt nincsenek szűréshez és kereséshez használt adatbázis funkciók, a teljesítmény kompenzálja ezt a hátrányt, hisz minden adat elérhető egy egyszerű elérési úttal.

Ebben az adatbázisban az imént leírt szempontokat figyelembevéve a következő adatokat tárolom:

- admin / logopédus beállításai
- alkalmazás funkcióival kapcsolatos globális beállítások
- feature flag-ek

Egy konkrét példa a logopédus által beállított munkaidő. Ez nem függ semmi mástól és minden felhasználónak ugyanazt kell látnia. Ezekre az adattípusokra a Realtime Database volt a jó választás.

3.6.5. Storage

A Firebase Storage egy biztonságos és megbízható felhőalapú tárhelyszolgáltatás, amely lehetővé teszi a képek, videók, dokumentumok és egyéb fájlok tárolását és elérését az interneten keresztül.

- Biztonság: A Firebase Storage a Google Cloud Platform biztonságos infrastruktúráján alapul, amely megvédi adatait a sérüléstől és a jogosulatlan hozzáféréstől.
- Megbízhatóság: A Firebase Storage globálisan elérhető, így biztos lehet benne, hogy adatai mindig elérhetőek lesznek, bárhol is legyen a világon.
- Teljesítmény: A Firebase Storage gyors és hatékony, így adatai gyorsan és egyszerűen elérhetőek lesznek.

A szakdolgozatomban a Firebase Storage-ot használom a kölcsönözhető fejlesztői eszközök és játékok képeinek tárolására. A képek a felhasználók számára fontos információt nyújtanak az eszközök és játékok megjelenéséről és funkcióiról. A Firebase Storage használatával biztosítani tudom, hogy a képek biztonságosan és megbízhatóan tárolódjanak, és hogy a felhasználók számára mindig elérhetőek legyenek.

3.7. PWA

Progressive Web App (PWA) egy modern webalkalmazásfejlesztési megközelítés, amelynek célja a webes alkalmazások teljesítményének és felhasználói élményének javítása, valamint az alkalmazások offline és mobilkészülékeken történő használatának lehetővé tétele. A PWA-k olyan webalkalmazások, amelyek a legújabb webes technológiákat és módszereket alkalmazzák annak érdekében, hogy a felhasználók számára olyan élményt nyújtsanak, amely közelít a natív alkalmazásokéhoz.

Az egyik fő jellemzője a PWA-knak a szolgáltatófüggetlen természetük, ami azt jelenti, hogy nem szükségesek specifikus alkalmazásboltokba történő feltöltéshez és letöltéshez. A felhasználók egyszerűen hozzáférhetnek hozzájuk a webböngészőjükből, és a PWA automatikusan alkalmazként települ és működik, akár offline is.

Az offline funkcionalitás a PWA-k egyik kulcsfontosságú vonása. A Service Worker nevű webes technológia lehetővé teszi az alkalmazásoknak, hogy az internetkapcsolat nélkül is működjenek, tárolják az adatokat és biztosítsák a felhasználóknak a korábban betöltött tartalmakhoz való hozzáférést. Ez jelentős előnyt nyújt olyan helyzetekben, ahol a kapcsolat instabil vagy egyáltalán nincs.

A PWA-knak további előnyei közé tartozik a gyors betöltési idő, az egyszerű hozzáférés a kezdőképernyőről, és a jobb teljesítmény különböző platformokon. Emellett támogatják

az értesítéseket, így az alkalmazások tudatában lehetnek a felhasználóknak fontos eseményekről még akkor is, ha éppen nem használják az alkalmazást.

A PWA-k alkalmazzák viselkednek, de fejlesztésük és karbantartásuk lényegesen egyszerűbb és költséghatékonyabb, mint a natív alkalmazásoké. A fejlesztők a PWA-k készítése során modernebb eszközökkel, például az Angular, React vagy Vue.js keretrendszerrel dolgoznak, és az alkalmazások könnyen skálázhatók és frissíthetők a webes technológiák folyamatos fejlődése mellett.

3.8. Google Chrome, mint fejlesztői eszköz

A Google Chrome a Google által fejlesztett ingyenes és népszerű böngésző, amely számos felhasználóbarát funkcióval és kifinomult technológiával rendelkezik. Elsődlegesen a sebességre, biztonságra és egyszerű használhatóságra összpontosítva vált az egyik legelterjedtebb internetböngészővé a világon.

A Chrome DevTools kiterjedt eszközkészletet kínál a fejlesztőknek, amely segítségével elemzik, hibakeresik és optimalizálják weboldalaikat. A böngészőben elérhető konzol, hálózati analitika, és egyéb fejlesztői eszközök segítik a hatékony webfejlesztést.

3.8.1. Lighthouse

A Lighthouse egy ingyenes, nyílt forráskódú eszköz, amely a Chrome böngészőben található, és weboldalak teljesítményét, használhatóságát, hozzáférhetőségét és más tényezőket értékel. A Lighthouse-t a Google fejlesztette ki, és beépítette a Chrome DevTools rendszerébe, így könnyedén elérhető és használható a böngészőből.

Fő részei:

- **Teljesítményértékelés:** A Lighthouse átfogó elemzést készít a weboldal teljesítményéről, beleértve a betöltési sebességet, a renderelési időt és egyéb teljesítményparamétereket. Ezen információk segítségével a fejlesztők pontosan láthatják, hogy mely részei az oldálnak okozhatnak lassúságot.
- **Használhatósági vizsgálatok:** Az eszköz ellenőrzi, hogy az oldal megfelel-e az alapvető használhatósági szabályoknak. Például értékeli a hiperhivatkozások megfelelő címkezését, a billentyűparancsokat, és egyebeket, amelyek hozzájárulnak a felhasználóbarát élményhez.

- Hozzáférhetőségi ellenőrzések: A Lighthouse ellenőrzi, hogy az oldal megfelel-e az akadálymentes webtervezési elveknek. Ebben az értékelésben figyelembe veszi például a kontrasztarányokat, a képek leírásait, és másokat, hogy biztosítsa az oldal hozzáférhetőségét különböző felhasználócsoporthoz számára.
- Progresszív webalkalmazás (PWA) értékelés: A Lighthouse ellenőrzi, hogy az oldal milyen mértékben felel meg a progresszív webalkalmazások kritériumainak. Megvizsgálja az offline használhatóságot, a gyorsbetöltési stratégiákat, és egyéb PWA-szempontokat.
- SEO (keresőoptimalizálás) elemzés: A Lighthouse kiértékeli az oldal keresőoptimalizáltságát is, figyelembe véve például a meta-címkék helyes használatát, az alt-teksteket és egyéb SEO-szempontokat.
- Nyomtathatóság: Az eszköz ellenőrzi, hogy az oldal mennyire alkalmas a nyomtatásra, és javaslatokat tesz annak érdekében, hogy az oldal tartalma a nyomtatott verzióban is jól látható és olvasható legyen.

3.8.2. React fejlesztői bővítmények

A fejlesztésnél a böngészőt sem hagytam ki berendezkedéskor a könnyebb munka érdekében. A Visual Studio Code programhoz hasonlóan egy böngészőt is testreszabhatunk egy szoftver projekt készítéséhez. A Chrome Web Store-ban több webalkalmazás fejlesztést segítő bővítményt is használtam a szakdolgozatom elkészítése során.

Például a React Developer Tools bővítményt. Ez a bővítmény a Facebook által fejlesztett, és a React alkalmazások hibakeresésére és elemzésére szolgál. A React Developer Tools lehetővé teszi a fejlesztők számára, hogy könnyen ellenőrizzék és manipulálják a React komponenseket, megvizsgálják azok állapotát és tulajdonságait, valamint gyorsan azonosítsák a problémákat az alkalmazások felhasználói felületén.

A Redux DevTools bővítményt is használtam, amely lehetővé teszi a Redux alapú alkalmazások állapotának, cselekvéseinek és reduktorainak követését és ellenőrzését. A bővítmény segítségével a fejlesztők könnyen visszajátszhatják a cselekvéseket, és láthatják az állapotváltozásokat, ami jelentősen segíti a hibakeresést és a fejlesztési folyamatot.

3.9. PWA kompatibilis operációs rendszerek

A webalkalmazást fejlesztés közben több operációs rendszeren és eszközön is teszteltem. A PWA technológiának köszönhetően a lejjebb említett operációs rendszereken közel natív élmény volt az alkalmazás használata.

3.9.1. Android

Az Android operációs rendszer a Google által fejlesztett és karbantartott mobilplatform, amely a legtöbb okostelefonon és tableten található. Az Android az egyik legelterjedtebb mobil operációs rendszer a világon, amely széleskörű alkalmazás-támogatással rendelkezik és folyamatos fejlesztések során igyekszik megfelelni a felhasználói igényeknek.

Az Android rendszer az open-source Android Open Source Project (AOSP) alapján épül, így számos gyártó saját módosításokkal és felületekkel rendelkező eszközöket is piacra dobhat. A rendszer számos szolgáltatást és funkciót kínál, mint például a gyors és sima felhasználói élmény, a sokoldalú alkalmazások támogatása, valamint a személyre szabható felhasználói felület.

A Progressive Web App (PWA) technológia az Android operációs rendszeren keresztül is kiemelkedően támogatott. A PWA-k olyan webalkalmazások, amelyek fejlett funkciókat és felhasználói élményt kínálnak, és képesek egyfajta "alkalmazásként" viselkedni. Az Android operációs rendszer lehetővé teszi a PWA-k installálását a kezdőképernyőre, és ikonként elérhetővé teszi őket, szinte úgy, mintha hagyományos alkalmazások lennének.

Az Android böngészői, például a Google Chrome, támogatják a Service Worker API-t és az App Manifestet, amelyek nélkülözhetetlenek a PWA-k funkcionalitásához. Emellett a PWA-k offline módjának, push értesítéseinek és egyéb fejlett funkcióknak való támogatás is része az Android rendszer funkcionalitásának. A PWA-k tehát a felhasználók számára kifinomult, gyors és rugalmas élményt nyújtanak az Android operációs rendszeren keresztül, és lehetővé teszik a fejlesztők számára, hogy egyszerűbben és hatékonyabban érjék el a széleskörű mobil felhasználói bázist.

3.9.2. iOS

Az iOS az Apple által kifejlesztett és karbantartott operációs rendszer, amely az iPhone, iPad és iPod Touch eszközökön fut. Az iOS rendszer kiválóan integrálja a hardvert és a szoftvert, és az Apple egyedi ökoszisztémáját alkotja. Az iOS kiemelkedik az egyszerű és intuitív felhasználói élményével, a magas biztonsági szabványaival és a folyamatosan frissülő alkalmazási környezetével.

Az App Store a iOS operációs rendszer központi alkalmazásboltja, ahol a felhasználók számos alkalmazást, játékot és szolgáltatást találnak. Az App Store a következetes minőségellenőrzések és biztonsági intézkedések miatt híres, és az alkalmazások letöltése és frissítése rendkívül egyszerű és biztonságos.

Ami a Progressive Web App (PWA) támogatást illeti, az iOS rendszer korábban szigorú korlátokat állított fel. Az iOS korábbi verziói csak limitáltan támogatták a Service Worker API-t, és nem engedélyezték a PWA-k ikonként való hozzáadását a kezdőképernyőhöz.

Az iOS 14 és újabb verziói azonban lényegesen javították a PWA-k támogatását. A Safari böngésző immár kezeli a Service Worker API-t és az App Manifestet, lehetővé téve a PWA-k offline funkcionalitását, valamint az ikonként való hozzáadásukat a kezdőképernyőre. Bár a PWA-k funkcionalitása az iOS rendszeren továbbra is kisebb mértékben terjed ki, mint az Android esetében, az Apple folyamatosan fejleszti a PWA-támogatást, és várhatóan a jövőben további javulásokat hoz majd ezen a téren.

Így az iOS operációs rendszer is egyre inkább nyitottá válik a PWA-k iránt, és a fejlesztők számára lehetőséget kínál a modern, webes technológiák használatára a felhasználói élmény optimalizálása érdekében az Apple eszközökön.

3.9.3. Windows

A Windows operációs rendszer a Microsoft által kifejlesztett és karbantartott platform, amely széles körben elterjedt asztali és laptop számítógépeken. A Windows rendszer kiválóan integrálja a hardvert és a szoftvert, és egy megbízható, felhasználóbarát környezetet kínál. A legfrissebb verziók, mint például a Windows 10, számos fejlesztést hoztak az élmény és a funkcionalitás terén.

A Microsoft Store a Windows operációs rendszer központi alkalmazásboltja, ahol a felhasználók számos alkalmazást, játékot és egyéb szoftvert találnak. Az alkalmazások

letöltése és frissítése egyszerű, és a Windows rendszer széles körű kompatibilitást kínál különböző alkalmazások és fejlesztési környezetek számára.

A Progressive Web App (PWA) támogatása a Windows rendszeren is egyre fontosabbá válik. A Microsoft a WebView2 alkalmazással lehetővé teszi a PWA-k futtatását a Windows rendszeren belül, és a következő Windows verziókban valószínűleg további fejlesztésekre számíthatunk a PWA-k támogatása terén.

Az Edge böngésző, amely a Windows 10-hez tartozó alapértelmezett böngésző, jelenleg is támogatja a Service Worker API-t és az App Manifestet, így a PWA-k egyszerűen telepíthetők és futtathatók az Edge böngészőn keresztül. Ezáltal a fejlesztők szélesebb körű célközönséghez juthatnak el, és lehetőségük van az asztali és laptop felhasználóknak is kényelmes és modern élményt biztosítani a PWA-k segítségével.

A Windows rendszeren a PWA-k támogatása fokozatosan bővül, és a Microsoft a webes technológiák felé nyitottabbá válik, lehetővé téve a fejlesztők számára, hogy az alkalmazásaikat széleskörűen elérhetővé tegyék az asztali felhasználók számára is.

4. Fejlesztői dokumentáció

A szoftver fejlesztésének célja egy modern, felhasználóbarát Logopédiai Időpontfoglaló és Eszközkölcsönző webalkalmazás létrehozása volt. A rendszer lehetővé teszi a logopédiai szolgáltatásokra történő online időpontfoglalást és eszközkölcsönzést egy egységes platformon keresztül.

4.1. Architektúra és technológiák

Ahogy a felhasznált technológiáknál említettem, a szakdolgozatomban számos szoftvert, keretrendszert, könyvtárt és fejlesztői eszközt felhasználtam. Ezek együttműködésével egy jól működő, stabil szoftvert hoztam létre.

4.1.1. Frontend

A frontend réteg fejlesztéséhez a React keretrendszert választottam. A React egy komponens alapú keretrendszer (bár könyvtárként is emlegetik), amely lehetővé teszi a könnyű és hatékony felhasználói felület kialakítását. A komponensek moduláris

kialakítása lehetővé teszi a könnyű újrafelhasználhatóságot és a fejlesztési folyamat gyorsítását.

A komponenseket 2 külön kategóriába soroltam, amelyek a mappaszerkezeten és a fájlnevezésen is látszódnak:

- oldalak (src/pages/<scope>/<name>Page.tsx)
- kis komponensek (src/components/<name>.tsx)

Az oldal komponensek a teljes oldalak, amelyekre az elérési útvonalak mutatnak. Csak az App.tsx fájlban vannak felhasználva. Az elérési utak és az URL paraméterek kezelését a react-router végzi. A webalkalmazásban található útvonalak a következők:

Útvonal	Komponens	Megjegyzés
index	MainPage	Főoldal, ami először tölt be, bejelentkezés logikát és a fő felhasználói felületet tartalmazza.
/booking	BookingPage	Időpontfoglalás
/admin	AdminPage	Admin felülete, csak megfelelő jogosultságokkal tölt be.
/rental	RentalPage	Eszköz kölcsönzés felülete.
/works/edit/ :workTitle	EditWorkPage	Foglalkozás szerkesztő az admin számára.
/appointments/edit/ :appointmentId	EditAppointmentPage	Időpont szerkesztő az admin számára.
/items/edit/:itemId	EditRentalItemPage	Kölcsönözhető eszköz szerkesztő az admin számára.

Minden oldal komponens gyökere egy Page komponens (src/components/Page.tsx), amely segíti a reszponzivitást, a fej- és lábléc elhelyezését és a konzisztencia fenntartását a webalkalmazás különböző helyein.

```
import { Container, Stack } from '@mui/material';
import React from 'react';
import Footer from './Footer';

function Page(props: any) {
  return <div>
    {props.header}
    <Container style={{paddingTop: 16}} maxWidth='md'>
      <Stack spacing={2}>
        {props.children}
      </Stack>
    </Container>
    <Footer />
  </div>;
}

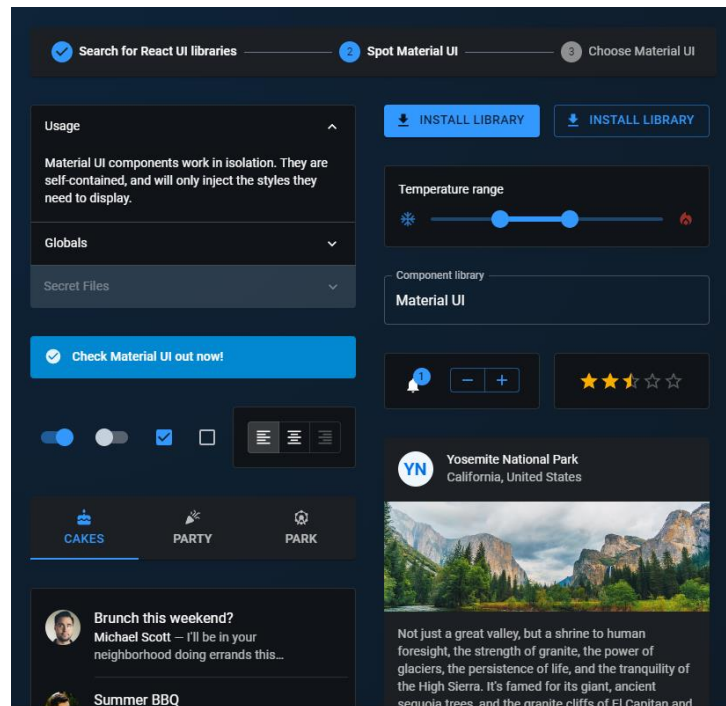
export default Page;
```

A frontend kevés, egyszerű komponensekből áll az átláthatóság és felhasználóbarát felület létrehozásához. Az oldal komponenseken kívül ezeket az építőelemeket használtam a felület elkészítéséhez:

Forrás	Komponens neve	Megjegyzés
saját	AppointmentDisplay	
saját	Footer	
saját	ItemDisplay	
saját	LoginCard	
saját	WorkDisplay	
@fullcalendar/react	FullCalendar	
@mui/material	AppBar, Button, Card, TextField és egyéb alap komponensek	

MUI komponens könyvtár, amely ismerős elemeket ad a webalkalmazásnak. Ez a komponens könyvtár a Google Material Design (2) tervezési nyelvét használja, amely 2014 óta az Android mobil operációs rendszer design-ja. Általában a saját projektjeimben is a Material Design tervezési nyelv irányelveit használom a felületek elkészítésekor. Így

egy szép, használható felületet gyorsabban tudok elkészíteni és több idő jut a backend és logikai részekre.



4.1.2. Backend

A webalkalmazás backend részét a Google működteti a Firebase szolgáltatásán keresztül.

- Authentication: Fiók kezelés, bejelentkezés e-mail jelszó párossal és Google fiókkal.
- Firestore Database: Adatbázis foglalkozásoknak, időpontoknak, kölcsönözhető fejlesztő eszközöknek és felhasználók adatainak.
- Realtime Database: Adatbázis az admin / logopédus beállításainak, globális beállításoknak az időpontfoglalással kapcsolatban és egyéb metaadatoknak.
- Storage: Fájlszerver a kölcsönözhető fejlesztő eszközök képei számára.
- Hosting: A webalkalmazás szolgáltatása a felhasználók számára.

Firestore Database és a Realtime Database a két fő adatbázis. Mindkét adatbázis NoSQL-adatbázis, ami azt jelenti, hogy nem táblázatos struktúrát használnak az adatok tárolására. Azonban számos kulcsfontosságú különbség van a kettő között, amelyeket figyelembe kell venni a döntés meghozatalakor.

- Realtime Database: A Realtime Database egy JSON-fa struktúrát használ az adatok tárolására. Ez egyszerűvé teszi az adatok tárolását és lekérését, de kevésbé

rugalmas, ha összetett kapcsolatokat kell tárolni. A Realtime Database az átvitt sáv szélesség és a tároló alapján kerül számlázásra.

- **Firestore Database:** A Firestore Database egy gyűjtemény-dokumentum-alcsoport modellt használ, amely lehetővé teszi a komplexebb hierarchiakat és a nagyobb lekérdezési rugalmasságot. A dokumentumok mezőket és beágyazott gyűjteményeket is tartalmazhatnak, ami ideális a strukturált adatok tárolására. A Firestore Database a végrehajtott műveletek (olvasás, írás, törlés) alapján kerül számlázásra, valamint a sáv szélesség és a tároló alapján is.

Ezeket a szempontokat figyelembe véve határoztam meg, hogy mely adatokat, melyik adatbázisba és milyen formában tároljam.

4.1.3. API

A Firebase szolgáltatások használata egy React alkalmazásból nem bonyolult feladat. A Firebase-nek hivatalos támogatása van a Node.js alapú projektekhez a firebase npm csomag formájában. A csomagon kívül szükség van még pár dologra. A „.env” fájlban meg kell adni az API kulcsokat, ez ad hozzáférést egy Firebase projekthez. Opcionálisan lehet egyéb beállításokat végezni a firebase.json fájlban.

Firestore használata React-ban:

```
import { initializeApp } from 'firebase/app';
import { getAnalytics } from "firebase/analytics";

const firebaseConfig = {
  // API kulcsok betöltése .env fájlból.
};

const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);

export { app };
```

- Az API kulcsok nem véletlen vannak kódon kívül. Ezeknek a kódba írása hatalmas biztonsági rés lenne. A „.env” fájl viszont csak a szerver oldalon létezik így nem érhetik el a kliensek a webapp forráskódjába nézve sem.
- Az app konstansban van minden, ami a Firebase projekt eléréséhez kell. Bár ezt exportálom a fájlból, nem használom a frontend-en, csak a Firebase-hez írt segéd függvényeimben.
- Az analytics konstans a Google Analytics szolgáltatás API-ja. Ennek a sornak köszönhetően tudom vizsgálni a webalkalmazásom látogatottságát a Firebase console-ban.

A különböző Firebase szolgáltatásokat külön helyekre rendeztem az átláthatóság érdekében a következő elérési úton: `src/firebase/<szolgáltatás>.ts`. Minden Firebase szolgáltatáshoz tartozó fájlban minden műveletnek külön függvénye van.

- `auth.ts`
 - `add/removeOnAuthStateChangedListener((User | null) => void)`
 - `getCurrentUser(): User | null`
 - `register(string, string)`
 - `login(string, string)`
 - `loginGoogle(boolean)`
 - `logout()`
- `firestore.ts`
 - `isAdmin((boolean) => void)`
 - `get/setUserEmail(string, string | (string) => void)`
 - `get/update/deleteWork`
 - `update/deleteAppointment`
 - `getAppointmentById`
 - `getAppointments(all)/ByDate/ByUser`
 - `getRentalItems`
 - `getRentalItemById`
 - `update/deleteRentalItem`
- `rttdb.ts`
 - `get/setAbout`
 - `get/setDayStart/End`
 - `get/setBreakBetweenWorks`

- storage.ts
 - uploadImage(File, (string) => void)

4.2. Fejlesztői környezet beállítása és projekt konfigurálása

A projekt helyi build-eléséhez a következő szoftverek szükségesek:

- Node.js (ajánlott: 18.16.0 vagy újabb)
- npm csomagkezelő
- VAGY Node.js és npm helyett Bun 1.0 vagy újabb, viszont a stabilitás érdekében javasolt a Node.js, npm párost használni

Nem szükséges, de ajánlott szoftverek:

- egy terminált tartalmazó és npm csomagkezelőt támogató kódszerkesztő vagy fejlesztői környezet (például Visual Studio Code vagy JetBrains WebStorm)

4.2.1. Projekt beállítása és build-elés

A PWA helyi build-elése a következő lépésekkel történik:

1. Nyisd meg a projekt mappáját (a mappa amely többek között tartalmazza a public és src mappákat és a package.json fájlt) terminálban vagy egy terminállal vagy npm támogatással rendelkező kód szerkesztőben vagy fejlesztői környezetben.
2. Add ki az npm install parancsot, amely telepíti a szakdolgozathoz szükséges npm csomagokat. Ha a mappában már létezik a node_modules mappa, erősen ajánlott az npm ci parancsot használni inkább, ami tiszta telepítést hajt végre függőségeken.
3. Indítsd el a build nevű npm script-et az npm támogatással rendelkező fejlesztői környezet vagy az npm run build terminál parancs segítségével.
4. Várj míg a folyamat véget ér. A kész build megtalálható a [projekt mappa]/build elérési úton. Rögtön ki is lehet próbálni az index.html fájl megnyitásával. A futtatáshoz internetkapcsolat szükséges és egy böngésző, amiben a JavaScript engedélyezve van.

A folyamat automatizálható is különböző verzió kezelő rendszerekben és egyéb környezetekben. Ilyen esetben érdemes ezt a parancsot használni:

```
npm ci && npm run build
```

A projekt GitHub repository-jában is van ilyen automatizálás. Minden merge-nél és pull request-nél történik egy build, majd sikeres lefutás után a projekt kiadásra kerül a Firebase Hosting szolgáltatásra. Merge-elni és pull request-eket elfogadni csak nekem van jogosultságom, tehát az automatikus build és deploy folyamatot csak én indíthatom el.

4.3. Adatrepresentáció

A webalkalmazásban a legtöbb adat tárolása és kezelése a Firebase szolgáltatáson történnek. A felhasználó autentikálásához, azonosításához, a globális beállításokhoz és fő funkciókhoz szükséges adatok mind egy Firebase adatbázisban vannak.

- Felhasználói adatok
 - Egyedi azonosító (id)
 - E-mail cím: A regisztrálásnál beírt vagy Google fiókból lekért e-mail cím.
 - Jelszó: E-mail-jelszó páros regisztráció esetén a felhasználó jelszava titkosítva.
 - Név: Google bejelentkezés esetén a felhasználó neve.
 - Profilkép URL: Google bejelentkezés esetén a felhasználó profilképének egy tömörített verziójára mutató URL.
- Időpontfoglalási adatok
 - Időpontok (appointments): Foglalt időpontok gyűjteménye. Tartalmazhat speciális elemeket, amelyek például szabadnapként jelenhetnek meg a felületen.
 - Egyedi azonosító (id)
 - Foglalkozás típusa (workTitle): A foglalkozás típusának neve.
 - Felhasználó azonosítója (userId): A felhasználó egyedi azonosítója, aki foglalta az időpontot.
 - Dátum (date): Egy YYYY-MM-dd formátumú dátum, ami a foglalkozás napját tárolja.
 - Idő (startTime): Egy HH:mm:ss formátumú idő, ami a foglalkozás kezdésének pontos idejét mutatja. Az időpont vége ebből a mezőből és a foglalkozás típusából (workTitle) kerül kiszámításra.

- Foglalkozások (works): Foglalkozások típusának gyűjteménye, amelyekre foglalható időpont.
 - Egyedi azonosító (id)
 - Cím (title): Foglalkozás neve.
 - Leírás (description): Foglalkozás hosszabb leírása.
 - Időtartam (durationMinutes): Foglalkozás hossza percben mérve.
- Eszköz kölcsönzési adatok. Egy eszköz adatai:
 - Egyedi azonosító (id)
 - Név (name): Kölcsönözhető eszköz neve.
 - Leírás (description): Eszköz leírása.
 - Kép URL-je (imageUrl): Egy URL, ami a tárgy képére vezet.
 - Kép (Firebase Storage-ban lévő fájl): Egy eszköz képe opcionálisan tárolható a webalkalmazás Firebase Storage adatbázisában. Ez bármilyen kép fájl lehet. Ebben az esetben az imageUrl mező értéke a storage-ban lévő fájl URL-je.
 - Jelenlegi érdeklődő E-mail címe (currentHolderEmail): Ebben a mezőben található a felhasználó e-mail címe, aki lefoglalta vagy kikölcsönözte az eszközt.
- Admin adatai és globális beállítások. Ezek az adatok mind a Firebase Realtime Database-ben találhatóak
 - Logopédus „Rólam” szövege (meta/about)
 - Munkaidő kezdete (appointment_settings/day_start): HH:mm formátumú idő.
 - Munkaidő vége (appointment_settings/day_end): HH:mm formátumú idő.
 - Segédérték az időpont foglalásnál használt delta érték számolásához, a logopédus preferált szünet ideje foglalkozások között (appointment_settings/break_between_works): HH:mm formátumú idő.

4.4. Időpont foglalás működése

Az időpontok foglalásához két adattípust hoztam létre:

- Work
 - Mezők: title, description, durationMinutes, tags

- Foglalkozás típusa és annak adatai. Ezeket lehet látni a kijelentkezett főoldalon és az időpont választó felületen, kivéve, ha a TAG_HIDDEN címkével rendelkezik. Erre a címkére a különleges esetek miatt van szükség. Ilyen például a szabadnap. A felhasználók nem választhatják foglalkozásként ezt, viszont az admin naptárában látszik.
- Appointment
 - Mezők: id, userId, workTitle, date, startTime
 - Maga az időpont és annak adatai. A userId mező az időpontot foglaló felhasználó azonosítója. A foglalkozások név szerint egyértelműen azonosíthatóak, így itt elég csak a nevüket eltárolni.

A szakdolgozatom elkészítése során figyelembe vettem a webalkalmazás fenntarthatóságát is. Mivel egy nem túl nagy célközönsége van a projektnek, ezért az olcsóbb lehetőségeket választottam néhány esetben. Ezért is történik a kliens oldalon a foglalható időintervallumok megállapítása. A Firebase Cloud Functions vagy React server component lehetőségekkel ezen lehet változtatni, ezzel az alkalmazást biztonságosabbá téve, viszont egyelőre ezt a megoldást elegendőnek tartottam.

Tehát a kliens végzi az időintervallumok számítását: Először lekéri az adatbázisból a releváns időpontokat és szabadnapokat. (Az elmúlt napok foglalásait feleslegesen nem kéri le.) A letöltött időpontok nem foglalhatók a felhasználó számára, viszont ezeken kívül is vannak foglalt időszakok. Az foglalkozások között egy bizonyos időszakot sem enged lefoglalni a webalkalmazás. Ez a „szünet” időtartam változó, a héten már foglalt időpontoktól és az adatbázisban lévő foglalkozás típusok hosszától függ.

4.5. Tárgy kölcsönzés működése

A kölcsönözhető tárgyak számára az ábrán látható osztályt készítettem. Minden eszköznek van azonosítója, neve, leírása, képe és állapota. Az állapottól függően egy e-mail cím is tartozik a tárgyhoz, ami a kölcsönzőhöz tartozik. Az eszköz állapotának egy saját típust hoztam létre a hibák elkerülése érdekében. TypeScript-ben a type kulcsszó segítségével hozhatunk létre saját típusokat, amelyeknek mi adhatjuk meg a megengedett értékeit. Hasonlóan működik, mint más nyelvekben az enumerátorok.

```
export type RentalItemStatus = 'available' | 'requested' |  
'rented' | 'unavailable';  
  
/*  
  available: item is available to be rented  
  requested: item has been requested by a user but not yet  
  approved so it can be requested by other users  
  rented: item is currently rented by a user and not available  
  unavailable: item is not available to be rented  
*/  
  
export default class RentalItem {  
  id: string;  
  name: string;  
  description: string;  
  imageUrl: string;  
  status: RentalItemStatus;  
  currentHolderEmail: string = '';  
  
  //constructor(){}  
}
```

4.6. Tesztelés és karbantartás

A webalkalmazás teszteléséhez több fizikai eszközt és szoftvert használtam. Mivel egy React webalkalmazásról van szó, rögtön a Chrome fejlesztői konzol eszközei jöhetnek szóba. Ezek közül a Lighthouse-ot emelném ki. Ezzel a szoftverrel teszteltem a webalkalmazás teljesítményét, adathasználatát és a PWA követelmények állapotát. Részletes értékelést ad a tesztelt projektről, így mindig tudtam mi hiányzik, hogy a PWA tökéletesen működjön.

A legtöbb embernek van mobilkészüléke és manapság népszerűbbé is vált, mint az asztali számítógépek. Ezért is terveztem a webalkalmazást elsődlegesen mobilhoz. Fejlesztés közben egyszerűen, a webalkalmazást megnyitva láthattam hogyan néz ki nagy képernyőn a felület és a Chrome fejlesztői eszközei között mobil nézet is található, viszont az alapos tesztelés érdekében fizikai eszközöket is használtam a felület

átnézéséhez. A Visual Studio Code-ba telepített Live server bővítmény segítségével a telefonjaimon is meg tudtam nyitni az éppen fejlesztés alatt lévő projektet. Tehát a webalkalmazás tesztelve van Windows, Android és iOS platformokon.

A React PWA-k karbantartása hasonló a hagyományos webalkalmazások karbantartásához. Azonban néhány szempontra kell figyelni, mivel a PWA-k telepíthetők a felhasználók eszközeire, és offline is használhatók. A hibajavításhoz ajánlott telepíteni a korábban említett Chrome és Visual Studio Code bővítményeket. A React általában nem túl olvasható hibaüzeneteket ad a fejlesztőnek. Viszont könnyebbségek is vannak egy natív alkalmazáshoz képest: Nincs áruház, ahová föl kell tölteni az alkalmazást. Így sokkal gyorsabban történhet egy frissítés kiadása. Egyedül a cache okozhat itt problémákat.

4.7. Továbbfejlesztési lehetőségek

A legtöbb projektem szinte sosem tekintem kész terméknek. Mindig lehet javítani, fejleszteni, egyszerűsíteni és szépíteni.

Az alkalmazás felhasználói felülete például könnyen továbbfejleszthető, hisz jelenleg az alkalmazás ilyen szempontból határozottan minimalista. Bőven van hely új funkcióknak. Felhasználó profil beállítások, jelszó átállítás, bejelentkezés egyéb szolgáltatásokkal, mint például Facebook vagy Apple fiókkal... Az időpontfoglalás algoritmus is bővíthet új paraméterekkel, ezáltal okosabbá válhat a funkció mikor a felhasználó jelentkezik egy foglalkozásra. A fókuszban nem a fejlesztő eszköz kölcsönzés funkció volt. Bár működik, itt elég sok lehetőség van újdonságokra. Részletesebb nézet a tárgyaknak, értesítések a tárgyak állapotáról, felhasználóbarátabb felület az adminnak a tárgyak kezeléséhez.

Az egyik legnagyobb továbbfejlesztési lehetőség az alkalmazáson belüli kommunikáció lenne. Az appon belüli chat implementálása segíthet a logopédusoknak és a pácienseknek egyszerűbben kommunikálni egymással. A chat rendszer lehetőséget adhat a kérdések, problémák megosztására, valamint segíthet az időpontok pontosításában és egyéb információk megbeszélésében. A chat funkció megvalósítása során szükséges a valós idejű kommunikációra alkalmas technológiát már használok a szakdolgozatban: Firebase Realtime Database. Ez az adatbázis, egyszerű, gyors és tökéletes választás egyszerűbb adatok, jelen esetben chat üzenetek tárolására. Viszont a továbbfejleszthetőség érdekében átgondolandó, hogy a Firestore Database jobb választás-

e. Ha több adatot szeretnénk tárolni egy üzenettel kapcsolatban, ez az adatbázis az előnyösebb.

Ha van kommunikáció, kellenek értesítések. Az értesítések bevezetése segíthet az időpontokról, eseményekről vagy egyéb fontos információkról történő gyors értesítésben. Ez javíthatja a felhasználói élményt és növelheti az alkalmazás használatát. Szerencsére egy PWA esetében van is lehetőségünk natív mobilértesítéseket küldeni a felhasználóknak.

5. Felhasználói dokumentáció

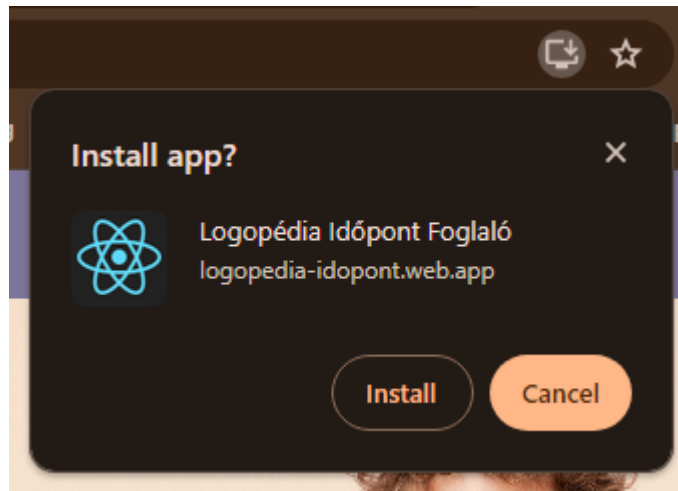
5.1. Webalkalmazás elérése és telepítés

A progresszív webalkalmazások látszólag nem különböznek sokkal egy egyszerű weboldaltól, hisz valójában az is. Ugyanúgy linkkel elérhető HTML-ből, CSS-ből és JavaScript-ből álló szoftver. A webalkalmazás a következő linken érhető el:

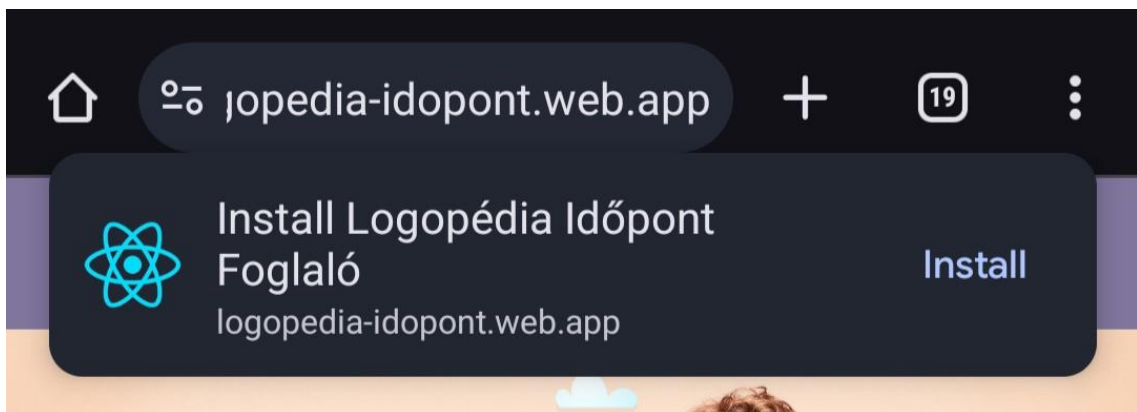
<https://logopedia-idopont.web.app/>

A webalkalmazás már használható is, rögtön a böngészőből. Viszont a natívabb élményhez és a jövőbeli egyszerűbb hozzáféréshez érdemes telepíteni az alkalmazást. Ez minden platformon eltérően néz ki, de hasonló a folyamat. Példák a 3 legnépszerűbb operációs rendszerhez:

- Windowson a PWA-kat a Microsoft Edge vagy Google Chrome böngészőből lehet telepíteni. A böngészőben nyisd meg a PWA weboldalát, és kattints a "Telepítés" gombra a böngésző címsorában. Jobb felső sarokban található, általában egy ikonként jelenik meg.



- iOS-en a PWA-kat a Safari böngészőből lehet telepíteni. A Safari böngészőben nyisd meg a PWA weboldalát, és koppints a "+" gombra a böngésző címsorában. Ezután érintsd meg a "Hozzáadás az otthoni képernyőhöz" lehetőséget.
- Android-on a PWA-kat a Chrome böngészővel lehet telepíteni. A böngészőben nyisd meg a PWA weboldalát és koppints a jobb felső sarokban a 3 pontos menüre. A menüben a "Hozzáadás a főképernyőhöz" gomb helyett egy telepítés gomb jelenik meg. Koppints arra a PWA telepítéséhez. Néhány esetben a rendszer automatikusan fölismeri egy weboldaltól, hogy telepíthető PWA. Ebben az esetben a böngésző magától felajánlja a telepítést:



5.2. Felhasználói felület és bejelentkezés

A webalkalmazás megnyitása böngészőben egy weboldalként, telepítve pedig splash screen-nel indul. Ezután megjelenik a fő felület. Akik használtak már Google szolgáltatást vagy Android-os telefonnal rendelkeznek, számukra rögtön ismerős lesz a

felület, hisz az alkalmazás a Material Design tervezési nyelv alapján készült. Egy leírás után lejjebb görgetve láthatóak a bejelentkezési lehetőségek és egy lista a szolgáltatásokról.

találkozzunk a következő időponton!
Üdvözlettel, Fanni

Időpont foglaláshoz és eszköz kölcsönzéshez jelentkezz be!

E-mail

Jelszó

REGISZTRÁLÁS

BEJELENTKEZÉS

BELÉPÉS GOOGLE FIÓKKAL

Szolgáltatások

Beszédindítás, nyelvi késés
terápiája ?


Vizsgálat, felmérés ?

Fiók létrehozására / bejelentkezésre több mód is van. Hagyományos e-mail jelszó páros és Google fiókos autentikálás. Fiókok összekapcsolására nincs lehetőség, tehát amelyik módot választja a felhasználó első belépésnél, azzal a móddal kell belépnie onnantól.

A regisztrálást vagy bejelentkezést követően 2 felület jelenhet meg: az admin és az ügyfél felülete. Mindegyik felület felső részén található egy kijelentkezés gomb és

mindegyik alján található egy lábléc a fejlesztő e-mail címével visszajelzések küldéséhez és segítség kéréséhez. Az ügyfél felülete:

Fanni Logopédia
KIJELENTKEZÉS



Fejlesztő eszközök, játékok és könyvek

BÖNGÉSZÉS

Foglalkozásaim

Apr 15 – 19
today
<
>

	Mon 4/15	Tue 4/16	Wed 4/17	Thu 4/18	Fri 4/19
6am					
7am					
8am					
9am					

+ ÚJ FOGLALKOZÁS IDŐPONT KÉRÉSE

Támogatás: jockahun@gmail.com

Ezen az oldalon minden fő funkció megtalálható. Az admin felületet később fejtem ki.

5.3. Időpont foglalás

Időpont foglalásához a felhasználónak a plusz ikonnal rendelkező “Új foglalkozás időpont kérése” gombra kell kattintania. Ekkor egy új oldalra kerül, ahol egyszerű lépésekben megadhatja a foglalkozást és az időpont dátumát, idejét.

1. Foglalkozás választása: a felhasználó a foglalkozás nevére kattintva választhatja ki a foglalkozást- A “?” ikonnal pedig megtekintheti a részleteit, mint például, hogy kinek, milyen korosztálynak, milyen problémákhoz ajánlott.
2. Dátum és idő választása: A következő lépésben egy egyszerű naptár felületen lehet időpontot választani. A munkaidőn kívül és már foglalt helyekre nem engedi a webalkalmazás az időpont foglalását. Ezek a felületen elszürkített és színes

területként jelennek meg a naptárban. Több szabály is van, hogy mi számít foglalt időnek, de a felhasználónak ezekről nem kell tudnia.

3. Foglalás: Az utolsó lépésben a felhasználó átnézheti a foglalást beküldés előtt és ellenőrizheti, hogy mindent jól választott-e. A foglalkozás neve és időpont ideje látszik a lépések mellett, illetve a 3. lépésben egy helyen, közvetlen a foglalás gomb fölött. Ezt a gombot lenyomva a webalkalmazás eltárolja a foglalást az adatbázisban, így megjelenik az admin felületen is és a felhasználó egy visszajelzést kap a foglalás sikerességéről.

Sikeres foglalás esetén a felhasználó egy kis felugró üzenetet kap és visszatér a főoldalára, ahol látja a foglalást a saját naptárában. Sikertelen foglalás esetén egy hibaüzenet jelenik meg és a foglaló oldalon marad, hogy újra tudjon próbálkozni a foglalással. Az időpontokat a főoldalon lévő naptárban le lehet mondani, ilyenkor az admin felületéről is törlődik az az időpont.

5.4. Tárgy kölcsönzés

5.5. Admin felület

6. Irodalomjegyzék

- [1] A. M. Magdolna, „Mit csinál egy logopédus?,” [Online]
<https://www.beszedrehabilitacio.hu/mit-csinal-egy-logopedus/>.
- [2] L. Szakszolgálat, „A logopédiai munka tevékenységi köre,” [Online]
<https://koraszulott.com/a-logopediai-munka-tevekenysegi-kore/>.
- [3] „Böngészőháborúk,” [Online]
<https://hu.wikipedia.org/wiki/B%C3%B6ng%C3%A9sz%C5%91h%C3%A1bor%C3%BAk>.
- [4] „Webböngésző,” [Online]
<https://hu.wikipedia.org/wiki/Webb%C3%B6ng%C3%A9sz%C5%91>.
- [5] L. Fanny, „Letölthető feladatok,” [Online] <https://logopediai-terapiak-veszpremben.webnode.hu/letoltheto-feladatok/>.

Mellékletek

Mappaszerkezet

```
+logopedia-dashboard
|   .env
|   .firebaserc
|   .gitignore
|   firebase.json
|   package-lock.json
|   package.json
|   README.md
|   tsconfig.json
+---build
|   \---[build fájlok]
+---docs
|   \---[dokumentáció és hozzá tartozó fájlok]
+---node_modules
|   \---[node/bun csomagok]
+---public
|   |   favicon.ico
|   |   index.html
|   |   logo192.png
|   |   logo512.png
|   |   manifest.json
|   |   robots.txt
|   \---service-worker.js
\---src
    |   App.css
    |   App.test.tsx
    |   App.tsx
    |   index.css
    |   index.tsx
    |   logo.svg
    |   react-app-env.d.ts
    |   reportWebVitals.ts
    |   setupTests.ts
    |   store.ts
    +---components
    |   |   AppointmentDisplay.tsx
    |   |   Footer.tsx
    |   |   ItemDisplay.tsx
    |   |   LoginCard.tsx
    |   |   Page.tsx
    |   \---WorkDisplay.tsx
    +---firebase
    |   |   auth.ts
    |   |   firebase.ts
    |   |   firestore.ts
    |   |   rtddb.ts
    |   \---storage.ts
    +---media
    |   \---banner.png
    +---model
    |   |   Appointment.ts
    |   |   RentalItem.ts
```

```
|    \---Work.ts
+---pages
|   +---admin
|   |   |   AdminPage.tsx
|   |   |   EditAppointmentPage.tsx
|   |   |   EditRentalItemPage.tsx
|   |   |   \---EditWrokPage.tsx
|   +---user
|       |   BookingPage.tsx
|       |   MainPage.tsx
|       |   \---RentalPage.tsx
\-----\---types
        \---images.d.ts
```