Pannon Egyetem

Műszaki Informatikai Kar

[Tanszék neve]

Üzemmérnök-informatikus BProf

**SZAK-/DIPLOMADOLGOZAT**

**Logopédia időpont foglaló webalkalmazás**

**Csáktornyai Ádám József**

Témavezető: Baumgartner János

Külső/belső konzulens: Baumgartner János

2023

*<<témavezető neve>>*

Tartalomjegyzék

[Bevezető 5](#_Toc133748685)

[1. Piac- és szakirodalmi kutatás 5](#_Toc133748686)

[1.1. A mobileszközök és mobilalkalmazások fejlődése 5](#_Toc133748687)

[1.2. A PWA (progresszív webalkalmazás) technológia elterjedése és előnyei 6](#_Toc133748688)

[1.3. Az Android és az iOS operációs rendszerek összehasonlítása 7](#_Toc133748689)

[1.4. A logopédia területén használt mobilalkalmazások felmérése 8](#_Toc133748690)

[2. A logopédiai folyamatok támogatását megvalósító alkalmazás problémadefiníciója 8](#_Toc133748691)

[2.1. Bevezetés a logopédiai folyamatokba 8](#_Toc133748692)

[2.2. A logopédiai folyamatok kihívásai az alkalmazás szempontjából 8](#_Toc133748693)

[2.3. Az alkalmazás céljai és követelményei 8](#_Toc133748694)

[2.4. Az alkalmazás használatának előnyei a felhasználók és a logopédus számára 8](#_Toc133748695)

[2.5. Használati esetek 8](#_Toc133748696)

[2.6. Az alkalmazás funkcióinak összefoglalása 8](#_Toc133748697)

[3. Technológiai háttér 8](#_Toc133748698)

[3.1. Git, GitHub és GitHub Desktop 8](#_Toc133748699)

[3.2. Visual Studio Code 9](#_Toc133748700)

[3.3. TypeScript 10](#_Toc133748701)

[3.4. React 11](#_Toc133748702)

[3.5. Firebase 11](#_Toc133748703)

[3.5.1. Autentikáció 11](#_Toc133748704)

[3.5.2. Host-olás 11](#_Toc133748705)

[3.5.3. Firestore NoSQL adatbázis 11](#_Toc133748706)

[3.5.4. Realtime JSON adatbázis 11](#_Toc133748707)

[3.6. PWA 11](#_Toc133748708)

[3.7. Google Chrome 11](#_Toc133748709)

[3.7.1. PWA kompatibilitás a különböző rendszereken 11](#_Toc133748710)

[3.7.2. Lighthouse 11](#_Toc133748711)

[3.8. Android 11](#_Toc133748712)

[3.8.1. Az operációs rendszerről általánosan 11](#_Toc133748713)

[3.8.2. A rendszer és a PWA-k 11](#_Toc133748714)

[3.9. iOS 11](#_Toc133748715)

[3.9.1. Az operációs rendszerről általánosan 11](#_Toc133748716)

[3.9.2. A rendszer és a PWA-k 11](#_Toc133748717)

[3.10. Windows 12](#_Toc133748718)

[3.10.1. Az operációs rendszerről általánosan 12](#_Toc133748719)

[3.10.2. A rendszer és a PWA-k 12](#_Toc133748720)

[4. Fejlesztői dokumentáció 12](#_Toc133748721)

[4.1. Build-elés 12](#_Toc133748722)

[4.2. Közreműködés 12](#_Toc133748723)

[4.3. Adatmodellek 12](#_Toc133748724)

[4.4. Főbb funkciók működése 12](#_Toc133748725)

[5. Felhasználói dokumentáció 12](#_Toc133748726)

[5.1. Felhasználói felület 12](#_Toc133748727)

[5.2. Használati útmutató 12](#_Toc133748728)

[5.3. Hibaelhárítás és támogatás 12](#_Toc133748729)

[6. Összefoglalás 12](#_Toc133748730)

[Hivatkozások 13](#_Toc133748731)

[Mellékletek 14](#_Toc133748732)

Bevezető

Az elmúlt évtizedekben az informatika robbanásszerű fejlődése számos területen átformálta az életünket, és az egészségügyi szektor sem maradt ki ebből a folyamatból. Az egyre több emberi erőforrást és anyagi forrást igénylő egészségügyi rendszerek hatékonyságának növelése és az ellátás minőségének javítása érdekében az egészségügyben is egyre inkább előtérbe kerültek az informatikai megoldások használata.

A szakdolgozatom célja egy olyan progresszív webalkalmazás fejlesztése, amely támogatja a logopédiai folyamatokat egy foglalkozás megszervezésétől a fejlesztő játékok kölcsönzéséig. Az alkalmazás lehetővé teszi a logopédus és a páciensei közötti kommunikáció javítását, az időpontok egyszerűbb egyeztetésével, és az egyéni igényekre szabott foglalkozások kialakításával, illetve megkönnyíti a gyógypedagógiai és logopédiai foglalkozásokhoz való eszközök beszerzését fejlesztő játék kölcsönzési lehetőségekkel, eszköz ajánlatokkal és ötletekkel.

Ebben a dokumentációban bemutatom a kutatásaim, magát az alkalmazást, a készítéséhez felhasznált technológiákat, az alkalmazást működés közben és a fejlesztés folyamatát, részleteit.

# Piac- és szakirodalmi kutatás

## A mobileszközök és mobilalkalmazások 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 ún. 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.

## A PWA (progresszív webalkalmazás) technológia elterjedése és előnyei

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 telepíthetőek 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 reszponzívak, és a felhasználók offline módban is használhatják őket, amely azok számára előnyös, akik korlátozott hozzáféréssel rendelkeznek az internethez. 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.

## Az Android és az iOS operációs rendszerek összehasonlítása

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#, JavaScipt, TypeScipt 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, 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 máson) futó progresszív webalkalmazás működésében és megjelenítésében.

## A logopédia területén használt mobilalkalmazások 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. De ezen kívül nem igazán látható más alkalmazása az informatikának, tehát a szakdolgozatom egy informatika szempontból még teljesen új területen fog működni.

# A logopédiai folyamatok támogatását megvalósító alkalmazás problémadefiníciója

## Bevezetés a logopédiai folyamatokba

## A logopédiai folyamatok kihívásai az alkalmazás szempontjából

## Az alkalmazás céljai és követelményei

## Az alkalmazás használatának előnyei a felhasználók és a logopédus számára

Ha egy magántanárral vagy bármilyen szolgáltatást nyújtó kisvállalkozóval beszélünk, az egyik legnagyobb nehézségként az időbeosztást is emlegetik. Több ismerősöm is van, aki a munkája megkönnyítése érdeképen időpont foglaló alkalmazást kezdett el használni. Ez nem csak a maga dolgát tette egyszerűbbé, az ügyfelei is pozitívan vélekedtek erről a változtatásról.

## Használati esetek

* Időpontfoglalás: Az alkalmazás fő funkciója az időpontfoglalás. Ezért igyekeztem, hogy a felhasználó számára ez a folyamat a lehető legegyszerűbb legyen.

1. A felhasználó megnyitja az alkalmazást tetszőleges felületen. (böngészőben vagy mobileszköze főképernyőjéről)
2. A felhasználó regisztrál vagy bejelentkezik attól függően, hogy van-e már fiókja.
3. A felhasználó kiválasztja az időpontfoglalás lehetőséget.
4. Időpontfoglalás
   1. A felhasználó először választ az elérhető foglalkozások közül. Csak azokat a foglalkozásokat látja, amire neki szüksége van.
   2. A felhasználó választ egy dátumot. Csak az a dátum kattintható, amelyik munkaidőben van, a logopédus nem vett ki szabadnapot és van azon a napon a felhasználó számára elérhető időpont.
   3. A felhasználó választ egy időpontot. Az alkalmazás csak azokat az időpontokat fogadja el, ami minden időbeosztási ellenőrzésnek megfelel. Például nem szabad ütközésnek lennie vagy két foglalkozásnak azonnal egymást követnie.
   4. A felhasználó az utolsó lépésben ellenőrzi a megadott adatokat. Ha megbizonyosodott a kiválasztott foglalkozás, dátum és idő helyességéről, kiválasztja a foglalás lehetőséget.
5. Foglalás után egy visszaigazoló üzenet jelenik meg. Helyes foglalás esetén visszanavigáljuk a felhasználót a főoldalra és ott mutat az alkalmazás egy „Sikeres foglalás!” üzenetet, hiba esetén a foglaló képernyőn marad a felhasználó és ott ír ki az alkalmazás egy hibaüzenetet. A hibaüzenet alatt a felhasználónak lehetősége újrapróbálkozni vagy visszamenni a főoldalra.

* Fejlesztő játék kölcsönzés
  1. A felhasználó megnyitja az alkalmazást tetszőleges felületen. (böngészőben vagy mobileszköze főképernyőjéről)
  2. A felhasználó regisztrál vagy bejelentkezik attól függően, hogy van-e már fiókja.
  3. A felhasználó megnyomja a „Fejlesztői eszközök, játékok és könyvek” cím alatt a „Böngészés” feliratú gombot.
  4. A felhasználó kiválasztja a kölcsönözni kívánt eszközt.
  5. A felhasználónak először lefoglalásra van lehetősége. Ha ezt választja akkor a tárgy ebben az állapotban marad egy megadott időtartamig. Ilyenkor csak ő kölcsönözheti ki azt az eszközt, más felhasználók nem tudnak sem kölcsönözni, sem foglalni.
  6. A felhasználó kiválasztja a kölcsönzés lehetőséget.

Ez a funkció még nincs elméletben készen, ezért a használati esetet a dokumentáció kész verziójában írom végig.

## Az alkalmazás funkcióinak összefoglalása

# Technológiai háttér

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.

## Git, GitHub és GitHub Desktop

A Git a legelterjedtebb verziókezelő rendszer a szakmában, sokan enélkül 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 biztonság érdekében a munkáim általában több helyen tárolom, biztonsági mentéseket készítek. Ebben 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, stb. Már középiskola eleje óta használom személyes és munkahelyi projekteknél és számtalanszor igénybe vettem az említett funkciókat.

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.

## Visual Studio Code

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

Minden projektem kezdetén, amelynél a VSCode-od használom, 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 projekthez hasznosnak talált néhány kiegészítő.

* VSCode hasznos beépített modulok
  + Emmet: gyors és egyszerű HTML, JSX és TSX szerkesztés
  + JavaScipt és TypeScript nyelv szerver: 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

## TypeScript

A szakdolgozatom elkészítéséhez a TypeScript programnyelvet választottam, amelyet típusos JavaScipt-ként emlegetnek. Ez egy fordított nyelv, mely JavaScipt-re fordul. Számtalan előnnyel rendelkezik a JS-hez képest. Ez főként a típusosságának köszönhető.

Míg a tiszta JavaScipt-ben a változóknak akár futásidőben is változat a típusa, TypeScipt nyelvben mindennek meg kell adni a típusát, amit a fejlesztői környezet be is tartat. Ez rengeteg hibalehetőséget szűr ki már a fejlesztés ideje alatt. Nem lehet null, aminél nem adtuk meg, egy változó csak egy típussal rendelkezhet és vannak új operátorok, műveletek, amelyek a típusok ellenőrzését, kezelését és null biztonságot teszik egyszerűbbé.

Egy másik jó tulajdonsága az, hogy bármelyik JavaScipt verzióra képes fordulni. Így nem kell aggódni, hogy mit támogat a futtató környezet. Legyen az egy böngésző, VSCode bővítmény vagy akár egy Node szerver.

E pozitív tulajdonságok mellett viszont hátrányok is vannak. A kód hosszabb lesz a típusok és sok ellenőrzés miatt. Illetve a kód nem futtatható önmagában, először fordítani kell. Utóbbi tulajdonság miatt így a szakdolgozatomban egy deploy ideje körülbelül 3 perc ahelyett, hogy csak egyszerűen feltölteném a fájlokat a host szerverre.

## React

A React a Meta (régen Facebook) által készített JavaScript keretrendszer, amellyel funkcionális úton készíthetünk felületet és frontend logikát egy webalkalmazásnak. A munka világában nagyon népszerű, ezért is szeretnék fejlődni ezzel a technológiával való munkában. Jelenlegi munkahelyemen is ezt használom banki webalkalmazások fejlesztéséhez.

## 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 máshoz.

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 találhatók példakódok minden hivatalosan támogatott programozási nyelvhez (Java, Kotlin, Swift, C# és még néhány).

### Autentikáció

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.

### Host-olás

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 szolgáltathatom 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 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.

### Firestore NoSQL adatbázis

Az iskolákban főleg az SQL adatbázisokat tanítják meg a diákoknak, pedig rengeteg más típus is létezik. Például a NoSQL adatbázis, ami a Firebase szolgáltatások egyik fő eleme. A Firebase SDK-t is szolgáltat a fejlesztőknek, amelyt egy projektbe implementálva hozzáférünk a választott programnyelvben az adatbázis kezelő függvényekhez. Például egy Android alkalmazás projektben egyszerű és beszédes Kotlin függvényekkel tudjuk az adatbázis elemeit lekérni, létrehozni, módosítani, szűrni és törölni.

Jelenlegi projektben TypeScript-tel történik az adatbáziskezelés és az adatok egyszerű JavaScript objektumként vannak jelen. Van lehetőség objektumok helyett osztályok használatára is és a szakdolgozatban ezt is teszem, hisz így tudom az adatbázisnál is igénybe venni a TypeScript nyelv előnyeit, mint például a típus ellenőrzés.

### Realtime JSON adatbázis

## PWA

## Google Chrome

### PWA kompatibilitás a különböző rendszereken

### Lighthouse

## Android

### Az operációs rendszerről általánosan

### A rendszer és a PWA-k

## iOS

### Az operációs rendszerről általánosan

### A rendszer és a PWA-k

## Windows

### Az operációs rendszerről általánosan

### A rendszer és a PWA-k

# Fejlesztői dokumentáció

## Build-elés

A projekt helyi build-eléséhez a következő szoftverek szükségesek:

* Node.js (ajánlott: 18.16.0 vagy újabb)
* npm csomagkezelő

Nem szükséges, de ajánlott szoftverek:

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

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.
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.

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 egy deploy a Firebase szolgáltatásra. Merge-elni és pull request-eket elfogadni csak nekem van jogosultságom, tehát az automatikus build és deploy folyamatot csak én indíthatom el.

## Adatmodellek

## Főbb funkciók működése

# Felhasználói dokumentáció

## Felhasználói felület

## Használati útmutató

## Hibaelhárítás és támogatás

# Összefoglalás

Hivatkozások

Mellékletek