**Szegedi Tudományegyetem**

**Informatikai Intézet**

**Szakdolgozat**

**Dobos Márk Dániel**

**2023**

**Szegedi Tudományegyetem**

**Informatikai Intézet**

**Webes felületű számítógép tervező alkalmazás**

**Szakdolgozat**

**Készítette:**

**Dobos Márk Dániel**

**Témavezető:**

**Dr. Bilicki Vilmos**

**Szeged**

**2023**

# Motiváció

A motivációm a ,,PC building” munkához kapcsolódóan a legfárasztóbb munkafolyamat megegyszerűsítéséből fakad. Minden munka előzménye a tervezés, ami ebben a műfajban egy sablonos unalmas folyamat, teli hibalehetőséggel.

Ennek alapjául a tervezési folyamatot fel lehetne gyorsítani egy alkalmazással, ami az alapvető hardvertulajdonságokat összehasonlítva meg tudja rövidíteni a tervezési folyamatot, illetve kizárni rengetek hibázási lehetőséget.

Saját célokra alkotom meg az alkalmazást, rövidtávuan a fenn említett előnyöket használnám ki, de később továbbfejlesztve a későbbi céljaim alapjául is szolgálni tudhat, egyéb funkciókkal kiegészítve.

# Feladatkiírás

A feladat egy számítógép összerakó webalkalmazás fejlesztése. A felhasználónak lehetősége van egy általa tetszőleges alkatrészekből számítógépet összerakni, valamint ezeket megrendelni. A megrendelt számítógépeket egy dolgozó megkapja, majd elbírálja, ellenőrzi, adott esetben elkészíti és teljesíti a rendelést.

# Tartalmi összefoglaló

## A téma megnevezése

A szakdolgozatom témája egy webes alapú alkalmazás implementálása Angular keretrendszer segítségével, melynek célja a témában dolgozó emberek munkájának meggyorsítása, leegyszerűsítése.

## A megadott feladat megfogalmazás

A feladat magába foglalja a projekt első részének implementálását, ami keretet ad egy nagyobb projekt és probléma jövőjének. Az alkalmazás platformfüggetlen, így bármilyen böngészésre alkalmas eszközről használható.

## Megoldási mód

A projektet az alkalmazáshoz hasonló piacon lévő megoldások elemzésével kezdtem, majd a sajátom megtervezésével folytattam, ami magába foglalja az adatfolyamatok, adatmodellek, képernyőtervek elkészítését.

## Alkalmazott eszközök, módszerek

A feladat megvalósításához, az Angulart használtam a felasználói felület elkészítéséhez. Backend pedig a Firebase segítségével működik, ahol a Firestore Database rendszerben tárolom az adatokat. A projekt tárolására pedig a GitHub verziókövető rendszert használtam. A fejlesztés a VS code fejlesztőkörnyezetbe történt.

## Elért eredmény

A szakdolgozatban sikerült megvalósítani több nélkülözhetetlen funkciót, illetve több nagyobb funkció alapját. Utat hagytam az alkalmazás továbbfejlesztésének, amit további funkciókkal fogják bővíteni az alkalmazást.

## Kulcsszavak

Angular, Firebase, számítógép építés, letisztultság

# Tartalomjegyzék

[Motiváció 3](#_Toc134881276)

[Feladatkiírás 3](#_Toc134881277)

[Tartalmi összefoglaló 4](#_Toc134881278)

[A téma megnevezése 4](#_Toc134881279)

[A megadott feladat megfogalmazás 4](#_Toc134881280)

[Megoldási mód 4](#_Toc134881281)

[Alkalmazott eszközök, módszerek 4](#_Toc134881282)

[Elért eredmény 4](#_Toc134881283)

[Kulcsszavak 4](#_Toc134881284)

[Tartalomjegyzék 5](#_Toc134881285)

[1.Területi áttekintés 7](#_Toc134881286)

[1.1. PCx: 7](#_Toc134881287)

[1.2. iPon: 7](#_Toc134881288)

[1.3. Összegzés 8](#_Toc134881289)

[2. Használt technológiák 8](#_Toc134881290)

[2.1 Angular 8](#_Toc134881291)

[2.2 Firebase 9](#_Toc134881292)

[2.3 Firestore Database 9](#_Toc134881293)

[3 Architektúrák 10](#_Toc134881294)

[4. Az alkalmazás szerkezeti felépítése 11](#_Toc134881295)

[4.1 Szolgáltatás (Service) 11](#_Toc134881296)

[4.2. Modulok 13](#_Toc134881297)

[4.3. Komponensek 13](#_Toc134881298)

[4.4. Routing 14](#_Toc134881299)

[4.5. Jogosultságkezelés 15](#_Toc134881300)

[5. Az alkalmazás használata, képernyőtervek 17](#_Toc134881301)

[5.1. Use-case diagram 17](#_Toc134881302)

[5.2. Bejelentkezés és regisztáció 18](#_Toc134881303)

[5.3. Felhasználók 20](#_Toc134881304)

[5.3.1. Egyszerű felhasználó 20](#_Toc134881305)

[5.3.2. Dolgozó felhasználó 21](#_Toc134881306)

[5.3.3. Admin felhasználó 22](#_Toc134881307)

[6. Adatmodell 22](#_Toc134881308)

[6.1. User kollekció 22](#_Toc134881309)

[6.2. News kollekció 22](#_Toc134881310)

[6.3. Computer kollekció 23](#_Toc134881311)

[6.4. Egyéb kollekciók 23](#_Toc134881312)

[7. A rendszer magas szintű áttekintése 24](#_Toc134881313)

[7.1. Egyed-esemény mátrix: 24](#_Toc134881314)

[7.2. Szerep-funkció mátrix: 24](#_Toc134881315)

[8. Fontosabb kódrészek és ezek ismertetése 25](#_Toc134881316)

[8.1. Regisztráció 25](#_Toc134881317)

[8.2. Bejelentkezés 25](#_Toc134881318)

[8.3. Számítógép elmentés 26](#_Toc134881320)

[8.4. Kijelentkezés 27](#_Toc134881321)

[9. Tesztelés 27](#_Toc134881322)

[10. Továbbfejlesztési lehetőségek 29](#_Toc134881323)

[Nyilatkozat 30](#_Toc134881324)

[Elektronikus melléletek 31](#_Toc134881325)

# **1.Területi** **áttekintés**

## 1.1. PCx:

PCX.hu egy magyarországi online áruház, amely széles választékot kínál a számítástechnikai termékek terén. Az áruházban találhatók laptopok, asztali számítógépek, tabletek, mobiltelefonok, perifériák, hardverek, szoftverek és egyéb elektronikai eszközök.

A PCX.hu az egyik legnagyobb számítástechnikai kereskedő Magyarországon, és a weboldaluk könnyen használható, így az ügyfelek gyorsan megtalálhatják a számukra megfelelő termékeket. Az oldalon számos szűrő lehetőség áll rendelkezésre, amelyek segítségével az ügyfelek szűkíthetik a termékpalettát az igényeiknek megfelelően.

A PCX.hu-nál az ügyfelek könnyen és biztonságosan vásárolhatnak online. Az oldal biztosítja a vásárlói adatok védelmét, és az áruk szállítása gyors és megbízható. Az ügyfélszolgálatuk szintén elérhető, hogy segítséget nyújtson a vásárlóknak, amikor csak szükségük van rá.

Összességében, a PCX.hu egy nagyon jó lehetőség a számítástechnikai termékek beszerzéséhez Magyarországon. A széles választék, a könnyen használható weboldal és az elérhető ügyfélszolgálat mind hozzájárulnak ahhoz, hogy az ügyfelek elégedettek legyenek a vásárlásukkal.

## 1.2. iPon:

Az iPon.hu egy magyarországi számítástechnikai áruház, amely széles választékot kínál az elektronikai eszközök terén. Az iPon.hu-n az ügyfeleknek lehetőségük van egyedi számítógépek összeállítására az általuk választott alkatrészekből, az iPon Gépösszerakó funkció segítségével.

Az iPon Gépösszerakó funkció lehetővé teszi az ügyfelek számára, hogy meghatározzák a számítógépük pontos konfigurációját. Az ügyfelek választhatnak a processzorok, alaplapok, memóriák, merevlemezek, SSD-k, videokártyák, tápegységek és más perifériák közül. Az iPon Gépösszerakó funkcióban az összes termék megfelelően kategorizálva található, így az ügyfelek könnyedén megtalálják azokat az alkatrészeket, amelyek megfelelnek a szükségleteiknek.

Az iPon Gépösszerakó funkció használatával az ügyfelek teljes ellenőrzést kapnak a számítógépük összeállításában. Az ügyfelek minden alkatrészre vonatkozó részletes információkat kapnak, és a rendszer automatikusan ellenőrzi a konfiguráció kompatibilitását. Az iPon Gépösszerakó funkció emellett lehetővé teszi az ügyfelek számára, hogy a kiválasztott alkatrészeket mentse, és később visszatérjen a konfigurációhoz.

Az iPon Gépösszerakó funkció előnye, hogy az ügyfelek egyedi számítógépeket készíthetnek el az általuk választott alkatrészekből, amelyek tökéletesen megfelelnek a szükségleteiknek. Emellett az iPon.hu szakértői technikusai is rendelkezésre állnak, hogy segítsenek az ügyfeleknek a számítógép összeállításában, vagy bármilyen egyéb kérdésükkel kapcsolatban.

## 1.3. Összegzés

A látottak alapján választottam ezt a szakdolgozat témát, igyekeztem számos nélkülözhetetlen funkciót megvalósítani, illetve úgy tervezni az adatbázisom, illetve a kódom, hogy teret adhassak a későbbi továbbfejlesztéseknek, hogy hasonló nagyszerű alkalmazás lehessen egyszer a piacon, mint a fenn elemzett kettő társa.

# 2. Használt technológiák

## 2.1 Angular

Angular egy open-source webes alkalmazásfejlesztő keretrendszer, amelyet a Google fejlesztett ki. A főbb tulajdonságai a következők:

Komponens alapú architektúra: Az Angular alkalmazások felépítése komponensekből áll, amelyek könnyen újrahasznosíthatóak és egymásba ágyazhatóak.

Kötés (binding) és direktívák: Az Angular lehetővé teszi a kétirányú adatkötést, ami azt jelenti, hogy ha az adat változik, akkor az automatikusan frissül a felhasználói felületen. Ezenkívül a direktívák segítségével a HTML kód könnyen manipulálható.

Szolgáltatások: Az Angular szolgáltatásokat használ az alkalmazás logikájának elválasztására a felhasználói felülettől. Ez lehetővé teszi a jobb moduláris felépítést és az egységtesztelést.

Moduláris felépítés: Az Angular moduláris felépítése lehetővé teszi a fejlesztők számára, hogy az alkalmazásukat kisebb darabokra bontva fejlesszék, ami csökkenti a hibalehetőségeket és megkönnyíti az újrafelhasználást.

Router: Az Angular router segítségével az alkalmazás lapjai közötti navigáció egyszerűen megvalósítható, az URL változásával történő automatikus frissítéssel együtt.

Dependency Injection (függőségkezelés): Az Angular függőségkezelő rendszere lehetővé teszi a szolgáltatások és más objektumok injektálását a komponensekbe, ami könnyebb teszi az egységtesztelést és az alkalmazás karbantartását.

## 2.2 Firebase

Firebase egy Google által készített platform, amely lehetővé teszi az alkalmazásfejlesztők számára az alkalmazások gyors és egyszerű fejlesztését és üzemeltetését. A főbb tulajdonságai a következők:

Adatbázis: Firebase lehetővé teszi az alkalmazásfejlesztők számára, hogy valós időben kezeljék az adatokat a Realtime Database segítségével.

Felhasználókezelés: A Firebase segítségével az alkalmazásfejlesztők könnyen kezelhetik a felhasználók azonosítását, hitelesítését és az engedélyezését.

Felhőalapú tárolás: A Firebase Cloud Storage lehetővé teszi az alkalmazásfejlesztők számára, hogy egyszerűen tárolják és kezeljék az alkalmazásukhoz kapcsolódó médiafájlokat.

Push értesítések: A Firebase lehetővé teszi az alkalmazásfejlesztők számára, hogy push értesítéseket küldjenek az alkalmazás felhasználóinak, amelyekkel azokat az alkalmazásba lehet irányítani.

Analytics: A Firebase segítségével az alkalmazásfejlesztők könnyen mérhetik az alkalmazásuk teljesítményét, nyomon követhetik az alkalmazás használatát és megérthetik a felhasználói viselkedést.

Machine Learning: A Firebase lehetővé teszi az alkalmazásfejlesztők számára, hogy a gépi tanulás segítségével javítsák az alkalmazásuk teljesítményét és szolgáltatásait.

## 2.3 Firestore Database

Firestore Database egy NoSQL dokumentumalapú adatbázis, amelyet a Google fejlesztett ki és a Firebase platform része. A főbb tulajdonságai a következők:

Skálázhatóság: Firestore lehetővé teszi az adatbázis horizontális skálázását, amely lehetővé teszi az alkalmazásfejlesztők számára, hogy az adatbázisukat egyszerűen bővítsék a felhasználói igényeknek megfelelően.

Valós idejű adatok: Firestore lehetővé teszi az alkalmazásfejlesztők számára, hogy valós időben figyeljék az adatokat és azok változásait, így az alkalmazásukban megjelenített adatok mindig naprakészek lesznek.

Dokumentumalapú adattárolás: Firestore lehetővé teszi az alkalmazásfejlesztők számára, hogy a dokumentumokban tárolják az adatokat, amelyek könnyen kezelhetők, módosíthatók és frissíthetők.

Tranzakciók: Firestore lehetővé teszi az alkalmazásfejlesztők számára, hogy tranzakciókat hajtsanak végre az adatbázisban, így az adatok konzisztensek és megfelelnek az üzleti logikának.

Indexelés: Firestore lehetővé teszi az alkalmazásfejlesztők számára, hogy az adatokat indexeljék és keresni tudjanak bennük, így az alkalmazásukban a felhasználók gyorsan megtalálhatják az általuk keresett információkat.

Többplatformos használat: Firestore lehetővé teszi az alkalmazásfejlesztők számára, hogy az adatbázist több platformon használják, például a weben, mobilalkalmazásokban és a szerveroldalon is.

# 3 Architektúrák

Egy Angular alkalmazásnak számos fontos eleme van, amelyeket a Firebase backend és a Firestore adatbázis használata tovább bővít. Néhány ilyen elem:

Futtatási környezetek: Az Angular alkalmazás futtatásához szükség van egy olyan futtatási környezetre, amely támogatja az alkalmazás által használt technológiákat, például a TypeScriptet és az RxJS-t. Az alkalmazás fejlesztésekor és tesztelésekor a fejlesztői szerver használata ajánlott, amely automatikusan frissíti az alkalmazást minden változtatás után.

Helyszínek: Az Angular alkalmazások többféle helyen futtathatók, például a felhasználók böngészőjében, asztali alkalmazásként vagy mobilalkalmazásként. Az alkalmazás tervezésekor és fejlesztésekor figyelembe kell venni a különböző helyszínek sajátosságait és korlátait.

Eszközök: Az Angular alkalmazásokhoz szükséges eszközök közé tartozik a fejlesztői környezet, például az Angular CLI, valamint a tesztelési és hibakeresési eszközök. A Firebase és Firestore használatához szükséges eszközök közé tartozik az adatbázis kezelői felület és az autentikációs eszközök.

Rétegek: Az Angular alkalmazások tipikusan több rétegből állnak, például az adatbázis rétegből, a szolgáltatás rétegből és a felhasználói felület rétegből. Az adatbázis rétegben a Firestore adatbázist használjuk, amely lehetővé teszi az adatok gyors és hatékony tárolását és lekérdezését. A szolgáltatás réteg felelős az adatok feldolgozásáért és az üzleti logika implementálásáért, míg a felhasználói felület rétegben megjelenítjük az adatokat és interakciókat biztosítunk a felhasználó számára.

Összefoglalva, egy Angular alkalmazásnak és a hozzá kapcsolódó Firebase backendnek és Firestore adatbázisnak számos fontos eleme van, amelyeket figyelembe kell venni az alkalmazás tervezése és fejlesztése során.

# 4. Az alkalmazás szerkezeti felépítése

## 4.1 Szolgáltatás (Service)

Az Angular keretrendszerben a service az egyik alapvető építőelem, amely nagyon fontos szerepet játszik az alkalmazás logikájának szervezésében és szétválasztásában. Az alábbiakban néhány okot sorolok fel, hogy miért fontos külön service-ket használni az Angular környezetben:

Kód újrafelhasználhatósága: Az Angular service-ek segítenek a kód újrafelhasználhatóságának javításában. A service-ek olyan kódrészletek, amelyeket több helyen is fel lehet használni, és amelyek lehetővé teszik, hogy az alkalmazás logikájának különböző részeit elkülönítsük és egymástól függetlenül kezeljük.

Moduláris kódstruktúra: A service-ek lehetővé teszik a moduláris kódstruktúra kialakítását, amely segíti a fejlesztőket abban, hogy az alkalmazás különböző részeit elkülönítsék egymástól. Ez a moduláris felépítés lehetővé teszi a fejlesztők számára, hogy az alkalmazás különböző részeit egyszerűbben és hatékonyabban fejlesszék.

Adatok és logika elkülönítése: A service-ek lehetővé teszik az adatok és a logika elkülönítését az alkalmazásban. Az adatszolgáltatás és a logika különválasztása lehetővé teszi, hogy az alkalmazás különböző részei egymástól függetlenül kezeljék az adatokat és a logikát.

Tesztelhetőség: Az Angular service-ek lehetővé teszik a kód egységtesztelését. Az egységtesztelési folyamat során a service-k külön tesztelhetők, ami lehetővé teszi a kód hibáinak korai felismerését és a javításukat, mielőtt azok problémákat okoznának az alkalmazásban.

Injektálás: Az Angular service-ek lehetővé teszik az injektálást, amely lehetővé teszi az alkalmazás egyes részeinek kommunikációját egymással. Az injektálás során az alkalmazás különböző részei közötti kommunikáció könnyen megvalósítható.

**Általam létrehozott szolgáltatások és metódusaik:**

**auth.service:**

* login() - felhasználó bejelentkezése
* signup() - felhasználó regisztrálása
* logout() – felhasználó kijelentkeztetése
* isUserLoggedIn() – aktuális bejelentkezett felhasználó lekérdezése

**computer.service**

* create() – számítógépp létrehozása
* getAll() – az összes számítógép lekérése
* geById() – egy darab lekérése *id* alapján
* update() – egy számítógép adatainak módosítása
* delete() – egy dokumentum törlése
* getPCbyStandardUserID() – a felhasználó összes elkészített számítógépének lekérdezése
* getPCbyWorkerUserID() – egy dolgozó által összes mukájának lekérdezése

**news.service**

* getAllNews() – összes hír lekérése
* delete() – egy hír törlése

**pcomponents.service**

Alkatrészek lekérdezésére szolgáló metódusok:

* getAllCPU()
* getAllGPU()
* getAllHDD()
* getAllMB()
* getAllPC\_case()
* getAllPSU()
* getAllRAM()

**user.service**

* getAll() – összes felhasználó lekérése
* getByID() – egy felhasználó lekérése
* delete() – egy felhasználó törlése

## 4.2. Modulok

Az Angular keretrendszerben a modulok az alkalmazás alapvető építőelemei közé tartoznak, és fontos szerepet játszanak az alkalmazás fejlesztése során.

**Általam használt/létrehozott modulok:**

* AppModule
* BrowserModule
* FormsModule
* CommonModule
* HttpClientModule
* ReactiveFormsModule
* RegisterModule, RegisterRoutingModule
* PcreateModule, PcreateRoutingModule
* LoginModule, LoginRoutingModule
* ListsModule, ListsRoutingModule
* HomeModule, HomeRoutingModule

## 4.3. Komponensek

* AppComponent
* AdminPageComponent
* WorkerComputersComponent
* StandardComputersComponent
* HeaderComponent
* FooterComponent

## 4.4. Routing

Az Angular is egy olyan keretrendszer, amely lehetővé teszi a Single Page Application (SPA) webalkalmazások fejlesztését, ahogy számos más keretrendszer is teszi. Az SPA weboldalak olyan egyoldalas webalkalmazások, amelyek dinamikusan frissítik az aktuális weboldal adatait a webszervertől származó új adatokra a felhasználói interakció során. Az SPA weboldalak eltérnek a hagyományos weboldalaktól, mert csak bizonyos részeit vagy részeit töltik be az oldalnak, nem pedig az egész oldalt vagy egy teljesen új oldalt vagy fájlt. Ezen részeket jelenítjük meg vagy rejthetjük el a felhasználó szeme elől, így olyan hatást keltve, mintha az egész alkalmazás egyetlen oldal lenne. A Routing ezt a célt szolgálja, amely segítségével navigálhatunk az oldalak között, és csak azokat a komponenseket jelenítjük meg a felhasználónak, amelyek szükségesek. Az Angular keretrendszer használata modulokon keresztül történik, amelyek biztosítják a kód strukturális szervezését és a függőségek kezelését.

A képen diagram látható

Automatikusan generált leírás

4.4.1 ábra – Routing diagram

Ezt a fajta navigációt URL-ek által valósítottam meg. Ha egy bizonyos oldalra navigál a felhasználó, akkor az útvonal megváltozik és az alkalmazásban a megadott oldal fog megjelenni.

## 4.5. Jogosultságkezelés

A jogosultságkezelés nagyon fontos egy webes alkalmazásban, mert segít biztosítani, hogy csak azok az emberek férjenek hozzá az alkalmazásban található információkhoz és funkciókhoz, akiknek joguk van hozzájuk. Ez megakadályozza a jogosulatlan felhasználók vagy hekkerek által okozott adatvesztést vagy károkat.

Például, ha egy webáruházban nem lenne megfelelő jogosultságkezelés, akkor egy betörő vagy rosszindulatú felhasználó könnyen hozzáférhetne az ügyfelek személyes adataihoz, mint például a név, cím vagy bankkártya információk. Ez nagy biztonsági kockázatot jelentene az ügyfelek számára és nagy károkat okozhatna az üzletnek.

A jogosultságkezelés lehetővé teszi továbbá, hogy az alkalmazás adminisztrátora korlátozza a felhasználók tevékenységeit. Például egy szociális hálózat alkalmazásban csak az adminisztrátoroknak kellene tudniuk törölni vagy módosítani a felhasználók bejegyzéseit. Ez megakadályozza, hogy az átlagos felhasználók véletlenül vagy szándékosan töröljék mások bejegyzéseit, ami sértő lehet.

Végül a jogosultságkezelés segíthet az alkalmazásban található információk védelmében is. Az alkalmazás adminisztrátora beállíthatja, hogy milyen szintű jogosultsággal rendelkező felhasználók férjenek hozzá az érzékeny információkhoz, például az üzleti titkokhoz vagy a védett személyes adatokhoz. Ez lehetővé teszi az adatvédelmi előírások betartását és az ügyfelek bizalmának megtartását.

Az Angular keretrendszerben több lehetőség van a jogosultságkezelés megvalósítására, amelyek közül a leggyakrabban használtak a következők:

1. Route Guard-ok: Az Angular alkalmazásokban a route guard-ok az útválasztási folyamatot vezérlő komponensek, amelyek megakadályozzák a felhasználókat abban, hogy olyan oldalakhoz férjenek hozzá, amelyekhez nincsenek jogosultságaik. A route guard-okat a "CanActivate" és "CanActivateChild" interfészek implementálásával lehet létrehozni, amelyek ellenőrzik, hogy a felhasználónak megvan-e a megfelelő jogosultsága az oldal eléréséhez.
2. AuthService: Az AuthService egy olyan szolgáltatás, amely segít az alkalmazásban található felhasználók azonosításában és azonosító tokenek kezelésében. Az auth service segítségével az alkalmazás különböző részein használhatunk ellenőrző függvényeket, amelyek ellenőrzik a felhasználó jogosultságát, és csak akkor engedik elérni a funkciókat, ha a felhasználó rendelkezik a megfelelő jogosultsággal.
3. Role-Based Authorization: A Role-Based Authorization (szerepkör alapú jogosultságkezelés) olyan módszer, amely lehetővé teszi az alkalmazás adminisztrátorainak, hogy különböző szerepköröket definiáljanak a felhasználók számára, és hogy az egyes szerepkörökhöz különböző jogosultságokat rendeljenek hozzá. Az alkalmazásban található különböző funkciókhoz való hozzáférés az adott felhasználó szerepkörének megfelelő jogosultságokkal van ellenőrizve.

Ezenkívül az Angular keretrendszer lehetővé teszi a token alapú autentikáció megvalósítását is, amely egy biztonságos és hatékony módja a felhasználók azonosításának és az alkalmazásban való hitelesítésének. Az Angular keretrendszer számos további lehetőséget is kínál a jogosultságkezelés megvalósítására, és az adott alkalmazás igényei szerint a fenti lehetőségeket kombinálhatja is.

Én az AuthService segítségéve oldottam meg az elérhető funkciók megjelenítését szerepkör szerint. Az adatbázisban létrehoztam egy *permission* nevűváltozót, ami három féle lehet:

* Általános felhasználó: 1-es
* Dolgozó felhasználó: 2-es
* Admin felhasználó: 3-as

Példának a weboldal fejlécén levő gombok megjelenését hozom, amit a 4.5.1-es kép szemlétet. Itt a header.comonent.ts-ben létrehozott *permission* nevű változóba az adatbázisból lekért jogosultság alapján jelennek meg a gombok vagy nem jelennek meg a gombok.

*4.5.1. ábra: header.component.html*

A 4.5.2-es képen pedig a *permission* változó értékadása látható a header.component.ts filean az ngOnInit() metódusban.

*A képen szöveg, képernyőkép, szoftver látható

Automatikusan generált leírás4.5.2. ábra: ngOnInit() metódus*

# 5. Az alkalmazás használata, képernyőtervek

## 5.1. Use-case diagram

Az alkalmazásban sok funkció van megvalósítva, amelyeket a use-case diagramon szemléltetek. Az alkalmazást minden szerepkörben limitáltan lehet használni.

A legtöbb funkció regisztrációhoz kötött, tehát egy egyszerű felhasználó a hírek listázásán kívül, illetve a másik szerepkörbe átlépésen kívül nem tud semmit csinálni.

Minden művelethez nélkülözhetetlen az adatbázissal való kapcsolat. Ez szükséges, mert az eszközön nem tárolunk adatokat és az alkalmazás sem használ konstans adatokat.

Az ábra is mutatja, hogy minden szerepkörnek meg vannak a saját funkciói és hozzáférései.

Az alkalmazáshoz még szükséges az internetelérés, enélkül nem fog működni az alkalmazás.

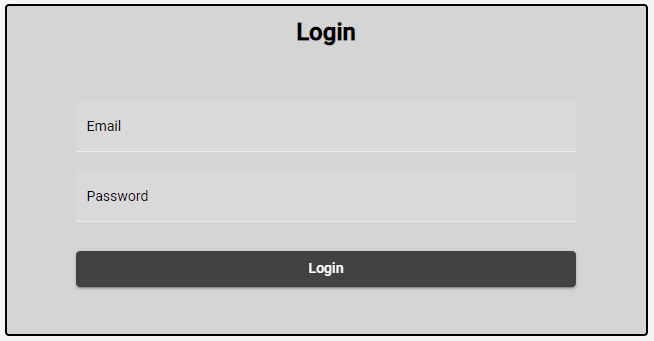
A képen diagram látható

Automatikusan generált leírás

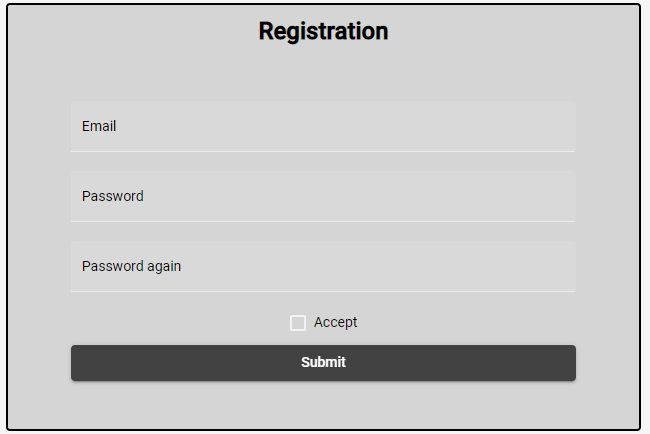
*5.1.1. ábra: use-case diagram*

## 5.2. Bejelentkezés és regisztáció

A bejelentkezést és a regisztrációt a Firebase egyik szolgáltatásának a Firebase Auth segítségével oldottam meg.



*5.2.1. ábra Bejelentkezés*

**

*5.2.2. ábra Regisztráció*

A Firebase szoftverfejlesztői csomagjában több függvény található. Egyik ilyen függvény a "createUserWithEmailAndPassword", amely az autentikációs példányt, a felhasználó e-mail címét és jelszavát várja paraméterként. Ez a függvény automatikusan regisztrálja a felhasználót a Firebase oldalon, eltárolva az utolsó bejelentkezés dátumát, profil létrejöttét, és egy saját felhasználói azonosítót ad az új profilnak. A "signInWithEmailAndPassword" függvény pedig bejelentkezéshez használható, szintén az e-mail címet és jelszót várja paraméterként. Ha az e-mail és jelszó helyes, az alkalmazás átnavigál a főképernyőre, ha pedig hibásak, akkor hibaüzenet jelenik meg. A mezők ellenőrzése Angular Validation segítségével történik, ahol különböző megkötések vannak, például a jelszónak legalább 6 karakter hosszúnak kell lennie, vagy a két különböző mezőben megadott jelszónak egyeznie kell. Ha a validátorok nem találnak hibát, akkor a megfelelő gomb aktiválódik és kattinthatóvá válik. Az egyes mezők hosszúsága, karakterszáma, kötelező kitöltési követelménye, vagy akár egyéni minta (pattern) is ellenőrizhető. Az e-mail mezőnél például egy regex mintát adhatunk meg a függvény paramétereként, amelyet a mezőt ellenőrző validátor alkalmaz. Ha a beírt cím megfelel a mintának, akkor a mező nem fog hibát jelezni. A hibaüzeneteket az aljára helyezett "ion-note" strukturális direktíva segítségével jelenítjük meg, amely csak akkor jelenik meg, ha az adott mező hibás feltételnek felel meg.

## 5.3. Felhasználók

Minden felhasználónak az elérhető funkciói a fejlécben található meg, amit ábrázolok a hozzá társított címszó alatt.

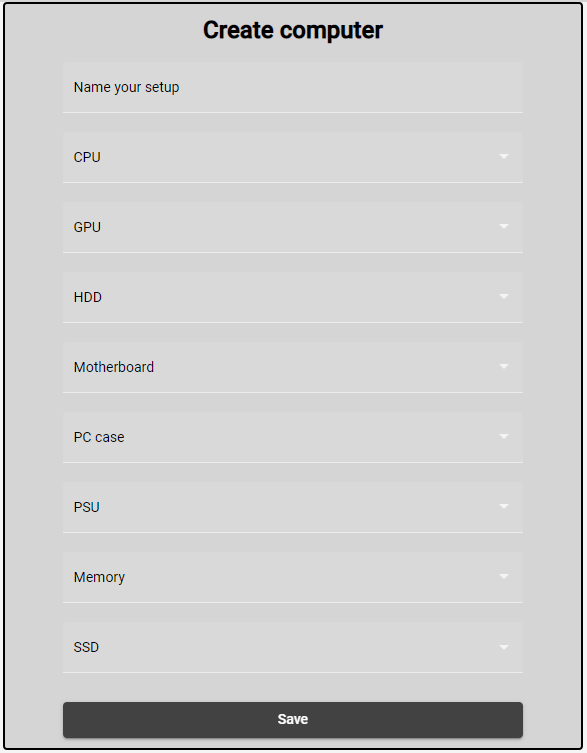
## 5.3.1. Egyszerű felhasználó

A képen szöveg látható

Automatikusan generált leírás

*5.3.1.1. ábra – Egyszerű felhasználó fejléce*

Az egyszerű felhasználó tudja listázni az általa készített komputereket, illetve tud is készíteni, ami a 5.3.1.2-es ábra szemléltet.



*5.3.1.2. ábra – A számítógép készítés formja*

## 5.3.2. Dolgozó felhasználó

A képen szöveg látható

Automatikusan generált leírás

*5.3.2.1. ábra – Dolgozó fejléc*

A dolgozó tudja listázni a megrendelt gépeket és tud nekik státuszt adni, aszerint, hogy elkészült-e, elutasításra került-e. (5.3.2.2. ábra)

A képen szöveg látható

Automatikusan generált leírás

*5.3.2.2. ábra – Megrendelések listázása*

## 5.3.3. Admin felhasználó

A képen szöveg látható

Automatikusan generált leírás

*5.3.3.1. ábra – Admin fejléc*

Az admin kezelheti helyileg az adatbázist:

A képen szöveg látható

Automatikusan generált leírás

*5.3.3.2. ábra – Admin felület*

Tud választani kollekciók közül, amit kiválaszt azt listázza és adott esetben tudja kezelni a kiválasztott dokumentumot.

# 6. Adatmodell

Az alábbi fejezetben az általam használt kollekciókat sorolom fel, melyek a következők:

## 6.1. User kollekció

A User kollekcióban tároltam el a felhasználó adatait, ide kerül be az összes regisztrál felhasználó, egységesen az admin, a dolgozó és az egyszerű felhasználó is.

|  |  |  |
| --- | --- | --- |
| id | a felhasználó id-ja | string |
| email | a felhasználó emailja | string |
| permission | a felhasználó hozzáférési kulcsa(1-standard, 2-dolgozó, 3-admin) | number |

*6.1.1. táblázat – User kollekció*

## 6.2. News kollekció

|  |  |  |
| --- | --- | --- |
| id | a hír id-ja | string |
| title | a hír címe | string |
| body | a hír érdemi része | string |

*6.2.1. táblázat – News kollekció*

## 6.3. Computer kollekció

|  |  |  |
| --- | --- | --- |
| id | a számítógép azonosítója | string |
| cpu | a benne található- processzor | string |
| gpu | -videókártya | string |
| hdd | -merevlemez | string |
| mb | -alaplap | string |
| pc\_case | -gépház | string |
| psu | -áramellátás | string |
| ram | -memória | string |
| ssd | -ssd | string |
| name | a felhasználó által választott mentési név | string |
| stage | az építés stádiuma | number |
| userStandardID | a készítő azonosítója | string |
| userWorkerID | a dolgozó azonosítója | string |

*6.3.1. – táblázat – A Computer kollekció*

## 6.4. Egyéb kollekciók

Minden számítógép komponens kapott egyéni kollekciót, hiszen mindnek vannak egyedi tulajdonságai is. Csak felsorolás szintjén:

* GPU kollekció: videókártya
* CPU kollekció: processzor
* HDD kollekció: merevlemez
* MB kollekció: alaplap
* PC\_case kollekció: gépház
* PSU kollekció: áramellátás (táp)
* RAM kollekció: memória
* SSD kollekció

# 7. A rendszer magas szintű áttekintése

A rendszer főbb folyamatai közé tartozik a bejelentkezés, kijelentkezés, regisztráció, hírek listázása, PC készítés, PC-k jóváhagyása-elutasítása, hírek- felhasználók listázása, törlése.

## 7.1. Egyed-esemény mátrix:

Az egyed-esemény mátrix az egyik alapvető eszköz az informatikai rendszerek tervezésében és dokumentálásában. Ez a mátrix segít az üzleti folyamatokat és az informatikai rendszer komponenseit összekapcsolni, és segít a tervezőknek megérteni, hogy a rendszer hogyan reagál az egyes felhasználói eseményekre. Az egyed-esemény mátrix fontos szerepet játszik az informatikai rendszer tervezési folyamatában (olvasás (O), módosítás (M), törlés (T), létrehozás (L)).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Bejelentkezés | Kijelentkezés | Regisztráció | Hírek | PC | Felhasználók |
| Látogató |  |  | L | O |  |  |
| Megrendelő | O | O |  | O | OLT |  |
| Dolgozó | O | O |  | O | OTM |  |
| *Admin* | *O* | *O* |  | *OT* |  | *OT* |

*7.1. táblázat – Egyed-esemény mátrix*

## 7.2. Szerep-funkció mátrix:

A szerep-funkció mátrix az egy másik fontos eszköz az informatikai rendszerek tervezésében és dokumentálásában. Ez a mátrix segít az üzleti szereplők és az informatikai rendszer szerepeinek és felelősségeinek azonosításában. A szerep-funkció mátrix fontos szerepet játszik az informatikai rendszer tervezési folyamatában

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Bejelentkezés | Kijelentkezés | Regisztráció | Hírek | PC | Felhasználók |
| Látogató |  |  | X | X |  |  |
| Megrendelő | X | X |  | X | X |  |
| Dolgozó | X | X |  | X | X |  |
| *Admin* | *X* | *X* |  | *X* |  | *X* |

*7.2. táblázat – Szerep-funkció mátrix*

# 8. Fontosabb kódrészek és ezek ismertetése

## 8.1. Regisztráció

A regisztráció egy Firebase szolgáltatáson keresztül valósítja meg a Firebase-el való

kommunikációt, ami a auth.service.ts fájlban valósul meg.

A képen szöveg látható

Automatikusan generált leírás*8.1.1. ábra*

Ezt a függvényt hívjuk meg a regisztrációhoz használt komponensünkbe, a

register.component.ts-be, itt végzünk a regisztrációs beviteli mezőkből egy vizsgálatot,

ami ellenőrzi, hogy a két jelszó egyezik-e.

A képen szöveg látható

Automatikusan generált leírás*8.1.2. ábra*

## 8.2. Bejelentkezés

A bejelentkezés hasonlóan a regisztrációhoz, a Firebase API híváson keresztül történik

meg, a megfelelő függvény definíció a auth.service.ts fájlban található meg:

A képen szöveg látható

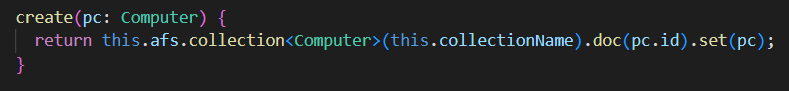
Automatikusan generált leírás*8.2.1. ábra*

A függvény implementációja pedig a login.component.ts-ben lelhető fel:

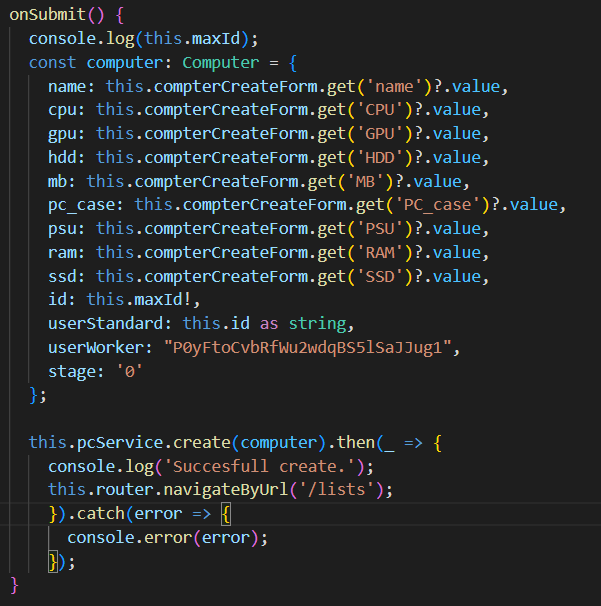
## A képen szöveg látható Automatikusan generált leírás*8.2.2. ábra*

## 8.3. Számítógép elmentés

A számítógép hozzáadását egy Firebase szolgáltatáson keresztül valósítja meg a Firebase-el való kommunikációt, ami a computer.service.ts fájlban valósul meg.

*8.3.1. ábra*

Ezt a függvényt hívjuk meg a pcreate.component.ts-ben:



*8.3.2. ábra*

## 8.4. Kijelentkezés

A Firebase megszakítje a bejelentkezés sessiont.

A képen szöveg látható

Automatikusan generált leírás

*8.4.1. ábra*

# 9. Tesztelés

A Webes alapú alkalmazások tesztelése rendkívül fontos, mivel segít biztosítani, hogy az alkalmazás a felhasználók számára biztonságos és megbízható legyen. Az alkalmazások tesztelése a fejlesztési folyamat során történik, és célja, hogy felfedje az esetleges hibákat és hiányosságokat az alkalmazásban, amelyek negatívan befolyásolhatják az alkalmazás teljesítményét vagy használhatóságát.

Az alkalmazás tesztelésének módszerei között szerepelnek a manuális tesztek és az automatizált tesztek. A manuális tesztek során a tesztelő személyenként teszteli az alkalmazást, és különféle teszteseteket hajt végre, hogy ellenőrizze az alkalmazás működését és az esetleges hibákat.

Az automatizált tesztek olyan tesztelési módszerek, amelyeket gépi eszközökkel végzünk. Ezek a tesztek sokkal hatékonyabbak és megbízhatóbbak lehetnek, mint a manuális tesztek, mivel az alkalmazásokat szabványos módon tesztelik, és az esetleges hibákat könnyen megtalálják.

Az alkalmazás tesztelése során a tesztelőknek figyelniük kell az alkalmazás működését különféle körülmények között, beleértve a különböző felhasználói interakciókat, a különböző böngészőket, az operációs rendszereket és a különböző eszközöket. Fontos, hogy a tesztelők alaposan dokumentálják a tesztelési folyamatot és a talált hibákat, hogy azokat a fejlesztők könnyen javíthassák.

Az alkalmazás tesztelése kulcsfontosságú része az alkalmazás fejlesztési folyamatának. Az alapos tesztelés segít csökkenteni az alkalmazásban lévő hibák számát, javítja az alkalmazás teljesítményét és megbízhatóságát, és biztosítja, hogy az alkalmazás a felhasználók számára biztonságos és használható legyen.

Az alkalmazás tesztelése során fontos figyelembe venni a különböző típusú teszteket is, amelyek segíthetnek a különböző funkciók és jellemzők tesztelésében. A tesztelési típusok közé tartozik a funkcionális tesztelés, az integrációs tesztelés, a teljesítménytesztelés, a biztonsági tesztelés és a kompatibilitási tesztelés.

A funkcionális tesztelés során a tesztelők az alkalmazás egyes funkcióit és jellemzőit tesztelik, például a regisztrációs folyamatot, a bejelentkezést vagy az adatok megjelenítését. Az integrációs tesztelés során az alkalmazást különböző rendszerekkel vagy szolgáltatásokkal integrálják és tesztelik, hogy azok helyesen működnek-e együtt. A teljesítménytesztelés során az alkalmazás teljesítményét tesztelik, beleértve a válaszidőt, az erőforrás-felhasználást és a terhelés alatti viselkedést. A biztonsági tesztelés során az alkalmazás biztonsági intézkedéseit tesztelik, például a jelszókezelést vagy a hozzáférési engedélyeket. A kompatibilitási tesztelés során az alkalmazás kompatibilitását tesztelik különböző böngészőkkel, operációs rendszerekkel és eszközökkel.

Az alkalmazás tesztelése során fontos, hogy az alkalmazás minden aspektusát alaposan teszteljük. Az alkalmazások tesztelésének hiánya vagy elégtelen tesztelési folyamatok jelentősen csökkenthetik az alkalmazás minőségét, ami végül a felhasználói elégedettséget csökkentheti. Ezért fontos, hogy az alkalmazások fejlesztése során körültekintően és alaposan teszteljük őket.

Az alkalmazást főként fejlesztés közben teszteltem a konzol segítségével, illetve utólag a formok újra kitöltésével.

# 10. Továbbfejlesztési lehetőségek

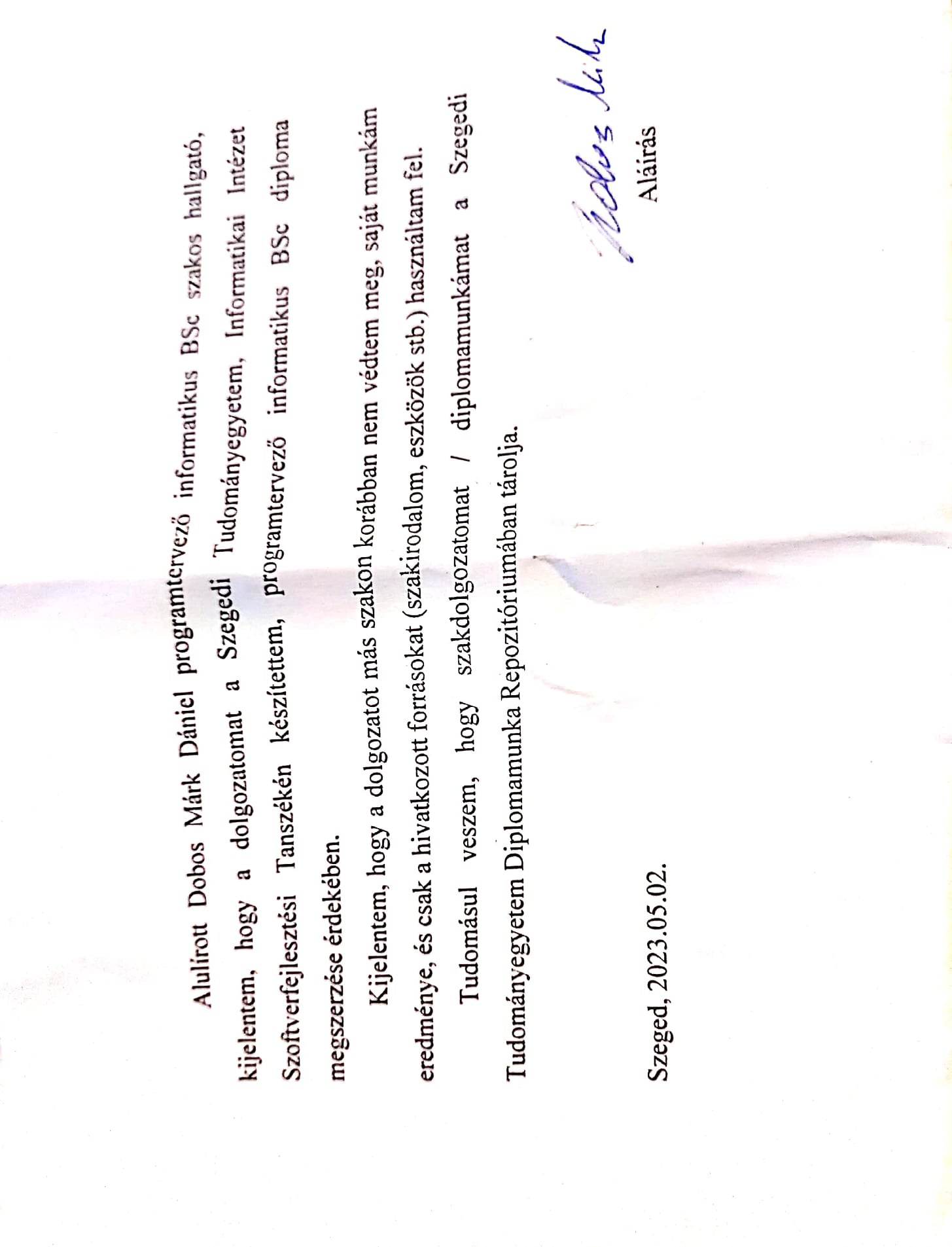
Az alkalmazás fejlesztése közben rengeteg tapasztalatra és tudásra tettem szert, ennek során számos új funkció és ötlet nyílt meg előttem. Megtanultam, hogy nem érdemes kellő tudás és körültekintés nélkül belevágni egy-egy funkció lefejlesztésébe, mert annak sok időt igénylő refaktorálás és újratervezés lesz az eredménye.

Minden tapasztalat alapján teljesen másképp kezdeném a szakdolgozat implementálását. Számos funkcióról lemondanék és helyettük másképp építeném fel a programot.

Alapvetően a program csak a szervizek munkájának meggyorsítására épül, egyszerű felhasználói lehetőségekkel kibővítve, mint a megrendelések. Akár lehetne még funkcióként hozzáadni, hogy a megrendelő és a dolgozó a felületen is tudjon kommunikálni, ne csak az alkalmazáson kívül eső csatornákon.

Egy dolgozó oldaláról is még hasznosabb lenne, ha tudná dokumentálni a munkáját, ne csak magának, hanem a munkatársainak is.

# Nyilatkozat



# Elektronikus melléletek

Az alkalmazás GitHub elérhetősége:

<https://github.com/Dobcsi26/Dobcsi26-Diplomamunka>

Alámondásos screencas elérhetősége:

<https://drive.google.com/drive/u/1/folders/1qiOakxqKEYqC0QCnctysB2EfyV0-WrXT>