Projektmunka dokumentáció

Barber Kereső

Premontrei Szakgimnázium és Technikum Keszthely

A projekt munkát ketten valósítottuk meg: Temleitner Marcell és Besze Marcell.

A mi általunk megálmodott alkalmazás neve Barber kereső.

Azért erre a témakörre esett a választásunk mivel csapatunk egyik tagjának nagy betekintése van a barber világában. Hosszas piackutás és felhasználói igények feltérképezése alapján realizáltuk, hogy a magyar piacon hiányzik egy olyan webalkalmazás ahol összegzi a fodrászatokat és fodrászokat. A különböző városokban információ hiány miatt a felhasználóknak gyakran nehézséget okoz a megfelelő szakember vagy szalon megtalálása, miközben a fodrászoknak is hiányzik egy központi platform, ahol könnyen elérhetik potenciális ügyfeleiket. A program egy frontend és backendből épül fel:

**Backend**: Ez a rész felelős a program funkcióiért, azaz az adatok kezeléséért, tárolásáért és feldolgozásáért. A backend kezeli a felhasználói regisztrációkat, a bejelentkezéseket, a szalonok és fodrászok adatainak tárolását, valamint az ügyfelek és fodrászok közötti interakciókat, például az időpontfoglalást. Az adatokat egy adatbázisban tárolja, és biztosítja a felhasználói kérések feldolgozását és a szükséges adatok küldését a frontendnek.

**Frontend**: A program megjelenítéséért felelős réteg, amelyet a felhasználók közvetlenül látnak és használnak. Itt valósulnak meg az interakciók, például a fodrászok keresése, időpontfoglalás, vélemények írása, valamint a felhasználói felület általános kezelése. A frontend biztosítja a felhasználóbarát felületet, ahol a vizuális elemek (gombok, űrlapok, listák) segítenek az információ könnyű megtalálásában és az alkalmazás használatában. A frontend kommunikál a backendel, és megjeleníti a feldolgozott adatokat a felhasználóknak.

**Projekthez használt programok:**

**VisualStudioCode(VSCode)**  
Kódfejlesztő környezet, amelyet különböző programozási nyelvekben való fejlesztések megvalósítására használható. Bővítményekkel és kiegészítőkkel nagyon testreszabható, így ideális webfejlesztéshez és projektekhez.

**Docker**  
Konténerizálási platform, amely lehetővé teszi, hogy alkalmazásokat futtassunk konténerekben, elszigetelve a környezettől. Ez segít biztosítani, hogy a programok minden gépen ugyanúgy működjenek, függetlenül az adott környezettől.

**BeekeeperStudio**  
Adatbázis-kezelő eszköz, amelyet relációs adatbázis amely a mi esetünkben PostgreSQL megtekintésére és szerkesztésére használható. Segít az adatbázis-lekérdezések futtatásában és kezelésében.

**Insomnia**  
REST API-k tesztelésére szolgáló eszköz. Használható API-hívások készítésére, fejlesztés közbeni tesztelésre, és könnyen nyomon követhetjük az API-válaszokat is.

**ReactDeveloperTools**  
Böngészőbővítmény, amely segít React-alapú webalkalmazások fejlesztésében és hibakeresésében. Különösen hasznos a React komponensfa és az állapotok nyomon követésére.

**GitHub**  
Verziókezelő platform, amely lehetővé teszi, hogy projektet közösségi formában kezeljünk, megosszunk, és követni tudjuk a változtatásokat. Git így fontos a csapatmunkához és az együttműködéshez.

**Discord**  
Kommunikációs platform, amelyet csapatmunka során használtunk. Támogatja a szöveges, hang- és videohívásokat is, így ideális projektmegbeszélésekhez és kollaborációhoz.

**Figma**  
Felhőalapú tervezőeszköz, amely grafikai tervek és prototípusok készítésére szolgál. Segítségével UI/UX terveket készíthettünk és együttműködhettünk a csapattal valós időben.

**PostgreSQL**  
Nyílt forráskódú relációs adatbázis-kezelő rendszer, amely strukturált adatokat tárol táblák formájában. Kiválóan alkalmas nagy teljesítményt igénylő adatbázisok kezelésére és komplex SQL lekérdezések végrehajtására, valamint támogatja a JSON típusú adatok kezelését is.

**Kihívások és Tanulságok a Fejlesztés Során**

A Barber Kereső fejlesztése során több kihívással is szembesültünk, amelyek különböző szempontokból tettek próbára minket, de egyúttal fontos tanulságokat is hoztak.

Az egyik legnagyobb kihívás az adatbázis-kezelés és az adatstruktúra tervezése volt. Mivel az alkalmazásban számos különböző típusú adatot kellett tárolnunk (felhasználói profilok, fodrászszalonok, időpontfoglalások, vélemények), komoly tervezést igényelt az adatbázis logikus és jól skálázható felépítése. A relációk közötti kapcsolatok megteremtése, például a felhasználók és a fodrászok közötti interakciók kezelése, kritikus pont volt a rendszer működése szempontjából. A tanulság az volt, hogy az adatbázis szerkezetének alapos megtervezése elengedhetetlen a hosszú távú fenntarthatóság érdekében. Megértettük, hogy a normalizálási szabályok következetes alkalmazása segít elkerülni a redundanciát és a későbbi teljesítményproblémákat.

A másik jelentős kihívás a felhasználói hitelesítés és biztonság kérdése volt. Egy olyan platform esetében, ahol a felhasználók személyes adatokat adnak meg, különösen fontos volt a megfelelő biztonsági intézkedések bevezetése. Ez nemcsak a jelszavak titkosított tárolását jelentette, hanem az adatbázis védelmét is, illetve az API-hívások biztonságos kezelését. A tanulság itt az volt, hogy a biztonsági elemeket már a fejlesztés korai szakaszában érdemes beépíteni, mivel ezek alapvetően befolyásolják az alkalmazás megbízhatóságát és a felhasználók bizalmát.

A frontend és backend kommunikációja szintén kihívást jelentett. A felhasználói felület interakcióit (pl. időpontfoglalás vagy fodrászok keresése) összekötni a háttérrendszerrel és az adatbázissal olyan technológiai kérdéseket vetett fel, amelyek alapos tervezést és tesztelést igényeltek. A tanulság ebből az volt, hogy az API-k pontos és megbízható implementálása, valamint a hibakezelés különösen fontos a felhasználói élmény szempontjából, hiszen minden lassulás vagy adatfeldolgozási hiba rontja az alkalmazás használhatóságát.

Összességében a projekt során számos technológiai és tervezési kihívással szembesültünk, de ezek segítettek minket abban, hogy nagyobb megértést szerezzünk az alkalmazások fejlesztésének kritikus pontjairól, különösen a biztonság, a hatékony adatkezelés és a skálázhatóság területén.

**Tesztelési és Hibakezelési Stratégiák**

A tesztelési és hibakezelési stratégiák kiemelt szerepet kaptak a Barber Kereső fejlesztése során, mivel fontos volt, hogy az alkalmazás minden funkciója hibamentesen működjön, és a felhasználói élmény zavartalan legyen. A tesztelés három fő aspektusra koncentrált: a unit tesztelés a backend logikai részének biztosítását, az API tesztelés a frontend és backend közötti kommunikáció helyességét, valamint a skálázhatósági tesztelés a rendszer hosszú távú stabilitásának és teljesítményének biztosítását célozta meg.

A unit tesztelés során a fő célunk az volt, hogy minden egyes funkcionális egységet (pl. felhasználói regisztráció, időpontfoglalás, szűrési logika) külön-külön teszteljük, hogy megbizonyosodjunk azok helyes működéséről. A Jest keretrendszert használtuk, mivel egyszerűen integrálható a JavaScript-alapú backend alkalmazásunkba, és biztosítja, hogy minden funkció külön-külön tesztelhető legyen. A tesztelés során kiemelt figyelmet fordítottunk arra, hogy a különböző logikai ágakat (pl. hibás adatbevitel, helytelen felhasználói művelet) is ellenőrizzük, ezzel biztosítva, hogy az alkalmazás ne csak a normál működés alatt, hanem a szélsőséges esetekben is megfelelően viselkedjen. A tesztelési eredmények alapján gyorsan lokalizáltuk és javítottuk a logikai hibákat, minimalizálva ezzel a későbbi problémák kialakulásának esélyét.

A API tesztelés elengedhetetlen volt ahhoz, hogy a frontend és backend közötti adatkommunikáció hibamentesen működjön. Erre a célra az Insomnia eszközt használtuk, amely lehetővé tette számunkra, hogy könnyedén teszteljük az API végpontokat. Az Insomnia segítségével API-hívásokat indítottunk a backend szerver felé, és ellenőriztük, hogy a várt válaszok helyesen érkeznek-e vissza. Különös figyelmet fordítottunk arra, hogy minden egyes végpont megfelelő HTTP státuszkóddal, az elvárt adatstruktúrával és a megfelelő válaszidővel működjön. Az Insomnia lehetőséget biztosított arra is, hogy minden API-hívást előre konfiguráljunk, és tesztelés során ismételten futtassunk, így könnyen azonosíthattuk a hibás válaszokat és a nem megfelelő működést. A hibás API-válaszok gyors feltárása lehetővé tette a backend kód optimalizálását és az API működésének finomhangolását.

A skálázhatósági tesztelés szintén fontos része volt a fejlesztésnek, mivel biztosítani szerettük volna, hogy az alkalmazás képes legyen kezelni a növekvő felhasználói igényeket és adatforgalmat, különösen a várhatóan megnövekedett látogatószámok és interakciók esetén. Ezt a tesztelést terheléses tesztekkel végeztük, amelyek segítségével szimuláltuk a magas forgalmat, és figyeltük a rendszer válaszidejét, erőforrás-használatát és teljesítményét. A tesztelés során például több ezer egyidejű felhasználói bejelentkezést és időpontfoglalást generáltunk, hogy meghatározzuk, hogyan reagál a rendszer a terhelésre. Az eredmények alapján optimalizáltuk a backend szervereket és az adatbázis lekérdezéseket, hogy a rendszer gyorsan és hatékonyan tudja kezelni a megnövekedett adatforgalmat, így biztosítva a skálázhatóságot és a hosszú távú stabilitást.

Összességében a unit tesztelés, az API tesztelés és a skálázhatósági tesztelés segítettek minket abban, hogy a Barber Kereső alkalmazás ne csak funkcionálisan, hanem teljesítmény szempontjából is optimálisan működjön. A tesztelési folyamatoknak köszönhetően biztosíthattuk, hogy az alkalmazás minden komponense megbízható, gyors és biztonságos, és képes lesz kezelni a jövőbeli felhasználói igényeket is.

**Biztonság és Felhasználói Adatok Védelme**  
A Barber Kereső alkalmazás kifejlesztése során kiemelt figyelmet fordítottunk a felhasználói adatok biztonságára és védelmére, hogy egy biztonságos és megbízható platformot nyújtsunk a fodrászok és ügyfelek számára. Mivel az alkalmazás olyan érzékeny adatokat kezel, mint például felhasználói profilok, foglalási információk és vélemények, elengedhetetlen volt, hogy minden adatvédelmi és biztonsági szempontnak megfeleljünk. A biztonság nem csupán technikai kihívásként, hanem a felhasználói bizalom alapvető feltételeként is szerepelt a projekt során. Az egyik legfontosabb szempont a felhasználói adatok védelme volt, különösen a bejelentkezési adatoké, mint a jelszavak. Az alkalmazásban **bcrypt algoritmust** alkalmaztunk a jelszavak titkosított tárolására, amely garantálja, hogy még egy esetleges adatbázis-kompromittálás esetén is gyakorlatilag lehetetlen visszafejteni a jelszavakat. A bcrypt segítségével minden jelszóhoz egyedi "salt"-ot adunk, amely tovább növeli a titkosítás biztonságát. A felhasználói jelszavak titkosítása nem csupán egy egyszerű hash-elési eljárás, hanem egy olyan módszer, amely az ismételt próbálkozásokkal szemben is ellenálló, így megnehezíti a brute force támadásokat.

### Fő funkciók

#### 1. **Fodrász keresése**

A felhasználók könnyedén kereshetnek fodrászokat a rendszerben:

* **Térkép megjelenítése**: Egy beépített térkép funkció az összes partner fodrász üzletet megjeleníti, így a felhasználók vizuálisan is könnyedén megtalálhatják a legközelebbi vagy preferált üzletet.
* **Szűrés fodrászok elérhetősége alapján**: A felhasználók látják, mely fodrászok elérhetők egy adott napon vagy időpontban, így egyszerűbbé válik a választás.

#### 2. **Időpontfoglalás**

Az alkalmazás lehetővé teszi az időpontok egyszerű foglalását:

* A felhasználó kiválaszthat egy fodrászt és időpontot, amikor a fodrász szabad.
* Az időpont foglalása során a felhasználók megadhatnak speciális megjegyzéseket vagy kéréseket.

#### 3. **Fodrász üzletek kezelése**

A fodrász üzletek (vagy fodrász szalonok) szintén megjelennek az adatbázisban:

* Az üzletekhez kapcsolódóan látható a nevük, címük, városuk, telefonos elérhetőségük, e-mail címük, valamint képek is tartozhatnak hozzájuk (pl. borítóképek vagy alapképek).
* Az üzletekhez kapcsolódnak a dolgozók (fodrászok), akik ott dolgoznak, és így az időpontokat is hozzájuk lehet foglalni.

#### 4. **Barátságok és Közösségi Funkciók**

* A felhasználók más felhasználókkal is baráti kapcsolatokat hozhatnak létre, ami segítheti a közösségi interakciót.

#### 5. **Fodrász elérhetőségének kezelése**

* A fodrászok az alkalmazáson keresztül megadhatják elérhetőségeiket (mely napokon és időpontokban dolgoznak), így az ügyfelek ezek alapján foglalhatnak időpontokat.
* Az elérhetőségi időpontok automatikusan frissülnek, amikor foglalás történik.

### Felhasználói út az alkalmazásban

Az alábbi lépéseken keresztül mutatom be egy tipikus felhasználó útját az alkalmazásban a regisztrációtól az időpontfoglalásig.

#### 1. **Regisztráció**

* A felhasználó először regisztrál az alkalmazásban. Ehhez meg kell adnia egy felhasználónevet, e-mail címet és jelszót.
* A felhasználó opcionálisan megadhatja a keresztnevét, vezetéknevét.
* A regisztráció során ki kell választania, hogy ügyfélként (client) vagy munkavállalóként (fodrász) szeretné használni az alkalmazást.

#### 2. **Bejelentkezés**

* A regisztráció után a felhasználó bejelentkezhet az e-mail címével és jelszavával, és hozzáfér az alkalmazás funkcióihoz.

#### 3. **Fodrász keresése**

* A bejelentkezést követően a felhasználó kereshet a különböző fodrászok között.
  + A felhasználó használhatja a térkép funkciót vagy kereshet konkrét **fodrász üzletek** alapján.
  + A térképes keresés különösen hasznos: a felhasználó látja az összes partner fodrász üzletet, és kiválaszthatja, hogy melyikhez szeretne időpontot foglalni.

#### 4. **Fodrász kiválasztása**

* Miután kiválasztotta a megfelelő fodrászt, a felhasználó ellenőrizheti a fodrász **elérhetőségét**. Az elérhető időpontok automatikusan frissülnek, ha más felhasználók foglalnak.
* A felhasználó megtekintheti a fodrász **profilját**, és esetleges különleges ajánlatait vagy szolgáltatásait.

#### 5. **Időpont foglalása**

* A felhasználó kiválasztja a kívánt **időpontot** és foglalást indít. A foglalás automatikusan megerősítésre kerül.
* Amennyiben szükséges, a felhasználó a foglalás során megadhat **speciális megjegyzéseket** vagy egyéb kéréséket a fodrász számára.

### Összefoglalás

Az alkalmazás egyszerű és felhasználóbarát módon segíti a felhasználókat abban, hogy fodrászokat találjanak, időpontot foglaljanak, és kapcsolatba lépjenek a szolgáltatókkal. A térképes keresési funkció kiemelten hasznos funkció, amely fokozzák a felhasználói élményt. Az egyértelmű lépések pedig garantálja, hogy a felhasználók könnyedén végigmenjenek az időpontfoglalás folyamatán.

**Adatbázis-terv és Adatszerkezet**

**1. User (Felhasználó)**

A User entitás a felhasználókat képviseli. Ez az alap entitás az alkalmazásban, és számos más entitáshoz kapcsolódik.

* Fő mezők: userId, username, email, password, firstName, lastName, profilePic.
* A role mezőben két típusú felhasználó létezik: client (ügyfél) és fodrász (munkavállaló).
* **Kapcsolatok**:
  + AvailabilityTimes: A felhasználó elérhetőségei.
  + Appointment: Foglalások ügyfelek és dolgozók között.
  + Friendship: Felhasználói kapcsolatok (barátságok).
  + StoreWorker: Az üzletek dolgozói és tulajdonosai.

Ez az entitás központi szerepet játszik a foglalások, üzletek és üzenetek kezelésében.

**2. Appointment (Időpontfoglalás)**

Az Appointment entitás az ügyfelek és dolgozók közötti időpontfoglalásokat reprezentálja.

* Fő mezők: appointmentId, status (lehetséges értékek: confirmed vagy completed), opcionális notes.
* **Kapcsolatok**:
  + User: Kapcsolat ügyfelek és dolgozók között (client, fodrász).
  + AvailabilityTimes: Az időpont elérhetősége (melyik időpontban történik a foglalás).

Az Appointment entitás lehetővé teszi az időpontok nyilvántartását.

**3. AvailabilityTimes (Elérhetőségi idők)**

Az AvailabilityTimes entitás tárolja a dolgozók elérhetőségeit.

* Fő mezők: timeSlotId, day, timeSlot, status (lehetséges értékek: accepted vagy available).
* **Kapcsolatok**:
  + User: Munkavállalókhoz kapcsolódik, akik megadják az elérhetőségi időpontjaikat.

Ez az entitás lehetővé teszi, hogy a dolgozók elérhetőségeit könnyen nyomon kövessék és időpontokat foglaljanak az ügyfelek.

Az üzenetküldő funkciót kezeli a chat szobákon belül.

**4. Friendship (Barátság)**

A Friendship entitás a felhasználók közötti baráti kapcsolatokat tárolja.

* Fő mezők: friendshipId, status (lehetséges értékek: pending, accepted, rejected).
* Kapcsolatok:
  + User: Két felhasználó között hoz létre kapcsolatot (user és friend).

Ez az entitás kezeli a felhasználói kapcsolatokat (barátkérések és státuszuk).

**5. Store (Üzlet)**

A Store entitás üzleteket reprezentál, ahol a dolgozók foglalásokat kezelhetnek.

* Fő mezők: storeId, name, description, address, city, postalCode, phone, email, latitude, longitude.
* Kapcsolatok:
  + StoreWorker: Az üzlethez tartozó dolgozók.

Ez az entitás az üzletek menedzselését támogatja, beleértve a helyszíneket és dolgozókat.

**6. StoreWorker (Üzlet dolgozója)**

A StoreWorker entitás az üzlet dolgozóit és tulajdonosait tárolja.

* Fő mezők: storeWorkerId, role (lehetséges értékek: owner, worker).
* Kapcsolatok:
  + Store: Az üzlet, ahol a dolgozó munkavégzést végez.
  + User: A dolgozó felhasználó.

Ez az entitás kezeli, hogy ki dolgozik vagy tulajdonos egy adott üzletben.

**Összefoglalás:**

Az adatbázis komplex és jól struktúrált, amely különböző funkciókat biztosít az alkalmazás számára. A legfontosabb kapcsolatok a User entitás köré épülnek, amely lehetővé teszi az időpontfoglalásokat, üzletek menedzselését, üzenetküldést, valamint a felhasználói kapcsolatok kezelését.

**Jövőbeli Fejlesztési Lehetőségek**

**1. Chat funkció**

* Miért hasznos?: A felhasználók és az adminisztrátorok közötti közvetlen kommunikáció gyorsítaná a foglalások intézését és a problémamegoldást. Ezen kívül a vendégek és a fodrászok között is lehetne beszélgetés a konkrét igények tisztázása érdekében (például hajstílus, időpontmódosítás stb.).
* Funkciók:
  + Barátok közötti chat: A barátnak jelölt felhasználók között könnyedén kommunikálhatnának egymással.
  + Szalonok és alkalmazottak közötti chat: Az alkalmazottak és a tulajdonosok egymás között is kommunikálhatnának, például a beosztásról vagy ügyféligényekről.
  + Ügyfél-szolgáltató chat: Ha valaki időpontot foglal, nyílhatna egy közvetlen üzenetváltási lehetőség a fodrásszal vagy a szalonnal.

**2. Értékelési rendszer és vélemények**

* Miért hasznos: Az ügyfelek értékelhetnék a szolgáltatásokat, és visszajelzést adhatnának az időpont foglalásaik után. Ez segíti a felhasználókat a szalonok közötti döntéshozatalban, és visszajelzést nyújt az alkalmazottaknak a munkájuk minőségéről.
* Funkciók:
  + 1-5 csillagos értékelési rendszer.
  + Szöveges vélemények hozzáadásának lehetősége.
  + Az fodrász moderálhatják a véleményeket, hogy biztosítsák a tisztességes visszajelzéseket.

**3. Értesítési rendszer (push értesítések)**

* Miért hasznos: Az értesítési rendszer segíthet az időpontok emlékeztetésében, a promóciók, akciók, vagy események gyors közlésében. Az admin felhasználók informálhatják az alkalmazottakat vagy a klienseket a fontos frissítésekről.
* Funkciók:
  + Időpont előtti értesítések (pl. 24 órával előtte emlékeztető).
  + Akciós ajánlatokról, új szolgáltatásokról való értesítések.
  + Rendszeres heti vagy havi emlékeztetők a szolgáltatásokról.

**4. Lojalitásprogram és kuponok**

* Miért hasznos: A visszatérő ügyfelek ösztönzése kulcsfontosságú a hosszú távú sikerhez. Egy lojalitásprogram, ahol például minden 5. látogatás után kedvezményt kap az ügyfél, növeli az elköteleződést.
* Funkciók:
  + Digitális hűségkártya: Minden alkalommal, amikor a felhasználó szolgáltatást vesz igénybe, egy "pecsétet" kap, és x számú pecsét után kedvezményhez jut.
  + Kuponok és promóciók: Az adminisztrátorok időszakos kuponokat hozhatnának létre, amelyek kedvezményeket biztosítanak az ügyfeleknek.

**5. Naptár integráció**

* Miért hasznos: Az időpontfoglalások könnyebb menedzselése érdekében a felhasználók szinkronizálhatnák az appot a saját naptáraikkal (Google Calendar, Apple Calendar stb.).
* Funkciók:
  + Automatikus naptárba történő esemény hozzáadása a foglalásokkor.
  + Foglalási emlékeztetők küldése az ügyfél naptárán keresztül.

**6. Szűrési opciók a szalonkereséshez**

* Miért hasznos: A felhasználók könnyebben megtalálhatják a számukra legmegfelelőbb szolgáltatót, ha a keresési lehetőségek bővítve vannak, például az ár, szolgáltatás típusa, helyszín vagy értékelés alapján.
* Funkciók:
  + Ár szerinti szűrés.
  + Szolgáltatás szerinti szűrés (pl. hajvágás, borotválkozás, festés).
  + Távolság szerinti szűrés (pl. a legközelebbi szalonok megjelenítése).

**7. Fényképes portfólió és galéria**

* Miért hasznos: A felhasználók szívesen nézegetik a fodrászok korábbi munkáit, hogy meggyőződjenek azok stílusáról és minőségéről. Egy galéria növelné az ügyfelek bizalmát.
* Funkciók:
  + A fodrászok és szalonok hozzáadhatnák a portfóliójukat az oldalhoz, ahol az előző munkáikat bemutathatnák.
  + Az ügyfelek is feltölthetnének előtte-utána képeket az értékelésük mellé.

**8. További fizetési opciók integrálása**

* Miért hasznos: Az online fizetési lehetőségek növelik a felhasználói kényelmet, és minimalizálják az elmaradt fizetéseket. Ez különösen hasznos lehet előre foglalt szolgáltatásoknál, ahol az ügyfél már előre tud fizetni.
* Funkciók:
  + Bankkártyás fizetés integrálása.
  + PayPal vagy egyéb online fizetési módok támogatása.
  + Előleg fizetése időpontfoglaláskor.

**9. Szalon profil személyre szabása**

* Miért hasznos: A szalonoknak lehetőséget adni arra, hogy saját profiljukat személyre szabják (például bemutatkozás, szolgáltatások, képek feltöltése), növeli az egyediséget és az ügyfelek meggyőzését.
* Funkciók:
  + Profilkép, borítókép hozzáadása.
  + Személyre szabott bemutatkozás és szolgáltatáslista.

**10. Több nyelv támogatása**

* Miért hasznos: A többnyelvű felület bővítené a felhasználói bázist, különösen, ha turisztikai célpontokon is elérhető a szolgáltatás. Ezzel több nemzetközi ügyfelet is elérhettek.
* Funkciók:
  + Automatikus vagy kézi nyelvváltás opciója.
  + Legyen legalább angol nyelv támogatva a nemzetközi felhasználók számára.

**Felhasználói Tesztek**

A felhasználói tesztek során különböző korosztályokból és háttérrel rendelkező emberek próbálták ki a Barber Appot, hogy felmérjük a funkcionalitást, felhasználói élményt és általános elégedettséget. Az alábbi fő eredmények születtek:

* Regisztrációs folyamat: A felhasználók túlnyomó többsége pozitívan értékelte a gyors és egyszerű regisztrációs folyamatot, de néhányan megjegyezték, hogy szívesen látnának több opciót, például Google- vagy Facebook-bejelentkezést, mivel ezek gyorsítanák a belépést.
* Felhasználói felület : A tesztelők nagyra értékelték az alkalmazás letisztult dizájnját és egyszerű navigációját, de néhány felhasználó jelezte, hogy a gombok és ikonok néhol túl kicsik vagy nehezen felismerhetők voltak.
* Időpontfoglalás funkció: A legtöbb felhasználó elégedett volt az időpontfoglalás könnyűségével és az elérhető borbélyok egyértelmű megjelenítésével. Azonban néhány tesztelő hiányolta a foglalások lemondásának rugalmasabb lehetőségét, különösen, ha közvetlenül aznapra szerettek volna időpontot foglalni vagy lemondani.

**2. Visszajelzések és Javasolt Fejlesztési Irányok**

A felhasználói visszajelzések alapján az alábbi javaslatokat tettük a fejlesztéshez:

* **UI/UX fejlesztések**:
  + Nagyobb ikonok és gombok: Az alkalmazás használhatóságának javítása érdekében érdemes lenne megnövelni az interaktív elemek méretét, hogy könnyebben kezelhetők legyenek, különösen kisebb képernyőkön.
  + Sötét mód bevezetése: Többen is javasolták a sötét mód bevezetését, ami kellemesebb használatot biztosíthat a felhasználóknak alacsony fényviszonyok között.
* **Időpontfoglalás optimalizálása**:
  + Rugalmasabb lemondási lehetőségek: Bevezetésre kerülhetne egy funkció, amely lehetővé teszi a felhasználóknak, hogy könnyebben lemondhassák vagy módosíthassák foglalásaikat az utolsó pillanatban, anélkül, hogy közvetlenül a borbélyhoz kelljen fordulniuk.
  + Várólista funkció: A felhasználók hasznosnak találnák, ha egy foglalt időpont felszabadulása esetén értesítést kapnának, vagy automatikusan bekerülhetnének egy várólistára.
* **Értesítések testreszabhatósága**:
  + A felhasználók számára hasznos lenne egy opció, amely lehetővé tenné, hogy kizárólag a foglalásokhoz és emlékeztetőkhöz kapcsolódó értesítéseket kapják, anélkül, hogy marketing- vagy promóciós anyagok érkeznének.
  + Ajánlások és értékelések: Az alkalmazáson belül elérhetővé válhatna egy felület, ahol a felhasználók megoszthatják tapasztalataikat, értékelhetik a borbélyokat és szolgáltatásokat, ami segíthet az új ügyfelek döntéseiben.

**3. Hosszú Távú Fejlesztési Irányok**

A rövid távú fejlesztési javaslatokon túl a felhasználói visszajelzések alapján a következő stratégiai irányvonalak körvonalazódtak:

* Lojalitásprogram: Bevezetésre kerülhetne egy lojalitásprogram, amely lehetővé tenné a felhasználók számára, hogy pontokat gyűjtsenek minden foglalás után, amelyeket később kedvezményekre válthatnának.
* Integráció más szolgáltatásokkal: Lehetőség nyílhat arra, hogy a Barber App összekapcsolódjon más szalonokkal és szolgáltatókkal, így a felhasználók egy platformon foglalhatnának különféle szépségápolási szolgáltatásokat, például fodrászatot, manikűrt vagy masszázst.

**NPM modulok**

**Backend**

**1. Fastify:**

A Fastify egy gyors és könnyen bővíthető web framework, amely kifejezetten az API-k és mikro-szolgáltatások fejlesztésére lett optimalizálva. Mivel performance-orientált, képes kezelni nagy terhelést is, miközben minimális erőforrást igényel. Az aszinkron kérések és válaszok gyors kezelése mellett a bővítmények (plugins) használata egyszerű, és a gyors validációval is rendelkezik.

* Használat: API végpontok és middleware-ek létrehozása, gyors HTTP kezelése.

**2. AWS SDK:**

Az AWS SDK for JavaScript egy eszközkészlet, amely lehetővé teszi, hogy Node.js alkalmazásokban integráljunk különböző felhőszolgáltatásokat. A Barber App például a használja a Cloudflare R2-t, képek tárolására.

* Használat: Felhőszolgáltatások elérése, például fájlok feltöltése a Cloudflare R2-ba vagy adatkezelés a Cloudflare adatbázis megoldásaival.

**3. pg (PostgreSQL):**

A pg modul a PostgreSQL adatbázissal való interakcióhoz szükséges. Az adatbázis-kezelés fontos a barber alkalmazás számára, például a felhasználók, időpontok, szolgáltatások és a barberek adatainak tárolására. A PostgreSQL egy megbízható, nagy teljesítményű, relációs adatbázis, amely ideális alkalmazásokat igénylő projektekhez.

* Használat: Adatok lekérése, beszúrása és módosítása PostgreSQL adatbázisban.

**4. TypeORM:**

A TypeORM egy ORM (Object Relational Mapping) könyvtár, amely lehetővé teszi, hogy a relációs adatbázisokat objektum-orientált módon kezeljük. A TypeORM lehetővé teszi, hogy a táblák és a rekordok helyett osztályokat és példányokat használjunk, ami segít a kód tisztaságának és könnyebb karbantartásának fenntartásában.

* Használat: Az adatbázisban való objektum-orientált munkavégzés.

**5. Zod:**

A Zod egy schema validáló könyvtár, amely különösen TypeScript-szel való integrációban erős. Ez segít a bemeneti adatok ellenőrzésében, így biztosítva, hogy a beérkező adat helyes és biztonságos legyen a backend számára. A Barber App esetén validáltuk a felhasználói regisztrációs adatokat vagy az időpontfoglalási űrlapokat.

* Használat: Bemeneti adatok validálása.

**6. TypeScript:**

A TypeScript az alapértelmezett nyelv, amely statikus típusellenőrzést biztosít a JavaScript felett. Segít elkerülni a futási időben jelentkező hibákat és jobban strukturálhatjuk az alkalmazást. A TypeScript ideális választás egy komplex alkalmazás, mint például a Barber App számára, mivel könnyebbé teszi a fejlesztést és a karbantartást.

* Használat: Típusellenőrzés, fejlesztési hibák elkerülése.

**7. Nodemon:**

A Nodemon egy fejlesztési eszköz, amely automatikusan újraindítja a Node.js alkalmazást, ha változások történnek a fájlokban. A Barber App fejlesztésekor segít, hogy ne kelljen kézzel újraindítani az alkalmazást minden egyes kódmódosítás után.

* Használat: Fejlesztési környezetben történő automatikus újraindítás.

**8. Jest:**

A Jest egy JavaScript/TypeScript tesztelő könyvtár, amely lehetővé teszi a gyors, hatékony és könnyen karbantartható tesztek írását. A Jest segít a fejlesztési hibák kiszűrésében és biztosítja, hogy az alkalmazás megbízhatóan működjön. A Barber App backendjén biztosítható a felhasználói regisztráció, időpontfoglalás és egyéb fontos funkciók tesztelése.

* Használat: Egység- és integrációs tesztelés.

**9. bcrypt:**

A bcrypt egy biztonságos jelszó-hashelő könyvtár, amely lehetővé teszi a felhasználói jelszavak titkosítását. A bcrypt használata elengedhetetlen, ha a Barber App felhasználói jelszavakat tárolnak és biztosítani kell azok védelmét.

* Használat: Jelszavak titkosítása és ellenőrzése.

Frontend

**1. DND (react-dnd):**

A react-dnd egy React könyvtár, amely lehetővé teszi a drag-and-drop funkciók egyszerű implementálását. A Barber App esetében például lehetőséget adhat a felhasználóknak, hogy egy szolgáltatást vagy időpontot áthúzzanak a naptárban, vagy akár a barberek közötti rendelési sorrendet is megváltoztathatják.

* Használat: Drag-and-drop interakciók implementálása, például időpontok átrendezése a naptárban.

**2. react-bubble-ui:**

A react-bubble-ui egy vizuális könyvtár, amely lehetővé teszi a buborékokkal való interakciók megvalósítását React alkalmazásokban. Ezt használhatjuk például a Barber App felületén, ahol különböző információkat, mint például a fodrász nevét, profilképét és hozzá tartozó időpontokat jeleníti meg buborékok formájában.

* Használat: Információs buborékok amik megjelenítik a fodrászokat és hozzá tartozó időpontjaikat.

**3. mapbox-gl:**

A mapbox-gl a Mapbox API-t használó JavaScript könyvtár, amely lehetővé teszi interaktív térképek megjelenítését a webalkalmazásban. A Barber App arra használja a térképet, hogy a felhasználók könnyen megtalálják a barbershop helyét, vagy akár az elérhető barberek helyszíneit.

* Használat: Interaktív térképek integrálása.

**4. AXIOS:**

Az Axios egy HTTP kliens, amelyet az API-k lekérésére és az adatküldésre használnak. A Barber App frontendjén az Axios lehetőséget biztosít a backend API-k elérésére, például az időpontfoglalások lekérésére vagy új felhasználók regisztrálására.

* Használat: HTTP kérések küldése és válaszok kezelése.

**5. js-cookie-jwt-decode:**

Ez a könyvtár lehetővé teszi a JWT (JSON Web Token) dekódolását, valamint a cookie-k kezelését a frontend alkalmazásokban. A Barber App esetében a felhasználói autentikációhoz és session kezeléshez használhatjuk a JWT tokeneket, például a bejelentkezés utáni token dekódolására.

* Használat: JWT token dekódolása és felhasználói adatok kezelése.

**6. React:**

A React az egyik legnépszerűbb JavaScript könyvtár, amelyet felhasználói felületek létrehozására használnak. A Barber App frontendje React alapú, mivel a könyvtár lehetővé teszi az interaktív, dinamikus komponensek és a felhasználói élmény kialakítását.

* Használat: UI komponensek létrehozása, alkalmazás állapotának kezelése.

**7. react-google-places-autocomplete:**

Ez a könyvtár lehetővé teszi a Google Places API használatát az autocomplete (automatikus kitöltés) funkcióval. A Barber App-ban a barber üzletek címének begépelésére szolgáló mezők automatikus kiegészítésére.

* Használat: Címek automatikus kiegészítése Google Places API-val.

**8. Tailwind CSS:**

A Tailwind CSS egy utility-first CSS framework, amely segít gyorsan és könnyedén reszponzív, testreszabott felhasználói felületek kialakításában. A Barber App frontendjén a Tailwind CSS segített abban, hogy a design gyorsan elkészüljön, miközben könnyen karbantartható és rugalmas marad.

* Használat: Stílusok gyors alkalmazása és testreszabása.