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 magyar 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 egyéb projektekhez.

**Docker**  
Konténerizálási platform, amely lehetővé teszi, hogy alkalmazásokat futtass 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 használhatsz. Segít az adatbázis-lekérdezések futtatásában és kezelésében.

**Insomnia**  
REST API-k tesztelésére szolgáló eszköz. Használható API-hívások készítésére, fejlesztés közbeni tesztelésre, és könnyen nyomon követheted 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 projekteket közösségi formában kezelj, megossz, és követni tudd a változtatásokat. Git alapú, így fontos a csapatmunkához és az együttműködéshez.

**Discord**  
Kommunikációs platform, amelyet csapatmunka során használható. 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 készíthetsz és együttműködhetsz 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 mélyebb 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.

 **Piackutatás és Felhasználói Igények**

* Ebben a szakaszban részletezhetnéd, hogyan és milyen módszerekkel végeztétek a piackutatást, milyen adatokat gyűjtöttetek a felhasználói igényekről, és mik voltak a legfontosabb megállapítások.

 **Funkcionalitás és Felhasználói Út**

* Leírhatnád részletesen a program fő funkcióit (pl. fodrász keresése, időpontfoglalás, értékelések, szűrés városok szerint). Emellett leírhatnád egy tipikus felhasználó "útját" az alkalmazásban a regisztrációtól az időpontfoglalásig.

 **Adatbázis-terv és Adatszerkezet**

* Részletezhetnéd, hogyan épül fel az adatbázis, milyen táblákat használtatok, és hogyan történik az adatok tárolása. Továbbá bemutathatnád az egyes kapcsolódó entitásokat (pl. felhasználók, fodrászok, időpontok).

 **Fejlesztési Eszközök és Technológiák**

* Bár már van egy lista a használt eszközökről, itt bővebben kifejthetnéd, miért választottátok ezeket az eszközöket, és hogyan segítették a fejlesztést (pl. miért PostgreSQL, miért Docker stb.).

 **Frontend és Backend Architektúra**

* Itt részletezhetnéd az alkalmazás szerkezeti felépítését. Bemutathatnád, hogyan kommunikál a frontend a backenddel (API hívások, adatfolyamok), és hogyan történik a különböző felhasználói interakciók kezelése.

 **Tesztelési és Hibakezelési Stratégiák**

* Leírhatnád, milyen módszerekkel teszteltétek az alkalmazást (pl. unit tesztelés, manuális tesztelés). Továbbá bemutathatnád, hogyan kezelitek a felmerülő hibákat és problémákat a fejlesztés során.

 **Biztonság és Felhasználói Adatok Védelme**

* Ebben a szakaszban bemutathatnád, milyen lépéseket tettetek a felhasználói adatok védelme érdekében (pl. jelszavak titkosítása, adatbázis-biztonság, hitelesítés).

 **Jövőbeli Fejlesztési Lehetőségek**

* Itt felsorolhatnád, milyen további funkciókat terveztek hozzáadni az alkalmazáshoz a jövőben (pl. mobilalkalmazás fejlesztése, új szűrési lehetőségek, további fodrászok hozzáadása).

 **Felhasználói Visszajelzések és Fejlesztési Irány**

* Kiemelhetnéd a felhasználói tesztek eredményeit, visszajelzéseket és ezek alapján tett javaslatokat a további fejlesztésekre.

 **Kihívások és Tanulságok a Fejlesztés Során**

* Leírhatnád, milyen nehézségekkel találkoztatok a projekt során, és hogyan oldottátok meg ezeket. Továbbá részletezhetnéd, milyen tapasztalatokat és tanulságokat szereztetek a folyamat során.