Vas Megyei Szakképzési Centrum

Nádasdy Tamás Technikum és Kollégium

PROJEKTFELADAT

TurboTech

Csonka Zoltán István, Oláh Márk

Konzulens:
Domnánovich Bálint

2025

Nyilatkozat

Alulírott, Csonka Zoltán István, Oláh Márk kijelentem, hogy a TurboTech című projektfeladat kidolgozása a saját munkám, abban csak a megjelölt forrásokat, és a megjelölt mértékben használtam fel, az idézés szabályainak megfelelően, a hivatkozások pontos megjelölésével.

Eredményeim saját munkán, számításokon, kutatáson, valós méréseken alapulnak, és a legjobb tudásom szerint hitelesek.

Csepreg, Szakony, 2025.04.01 _____

Hallgató

_____

Hallgató

Kivonat

TurboTech

Jelen záródolgozat egy olyan programot mutat be, amely könnyű segítséget nyújt a titkárok és a szerelők közti kapcsolat fenntartásában. A TurboTech egy olyan könnyen kezelhető modern alkalmazás, amely lehetővé teszi a gyors kapcsolat teremtést.

A programunk egy könnyű felhasználói felülettel rendelkezik ezzel elősegítve azokat a szerelőket vagy titkárokat, akik nem értenek 100 százalékban a technológiához. A program segítséget nyújt a titkárok számára, hogy gyorsan és könnyen feltudják venni a kapcsolatot a szerelőkkel. Segít a titkárnak a gyors munkafelvételben, gyors munkalezárásban.

Ezek mellett segítséget nyújt a szerelőknek, olyan módon, hogy könnyen megtekinthetik a munkáikat, amelyek függőben vannak vagy akár azokat is amelyek már lezárultak. Ez mellett maga a szerelő is letud zárni könnyen egy megadott munkát.

Abstract

TurboTech

This final thesis presents a program that provides easy help in maintaining the relationship between secretaries and mechanics. TurboTech is an easy-to-use modern application that allows you to make a quick connection.

Our program has an easy-to-use interface, helping mechanics or secretaries who are not 100 percent tech-savvy. The program helps secretaries to get in touch with mechanics quickly and easily. It helps the secretary to hire work quickly and close work quickly.

In addition, it provides assistance to mechanics in such a way that they can easily view their work that is pending or even those that have already been completed. In addition, the mechanic himself can easily close a given job.

Tartalomjegyzék

1. Bevezetés

A 2025-ös záróvizsgánk témájának egy olyan webes alkalmazást választottunk, amely lehetőséget nyújt a titkárok és autószerelők közötti gyors kapcsolat kiépítésére.

Fő célunk a porgrammal az volt, hogy egy olyan alkalmazást hozzunk létre, amely mind a titkárok számára mind az autószerelők számára könnyen kezelhető legyen, és gyors legyen.

Titkárként lehetőségünk van felvenni az ügyfél adatait, amelyet el tárolunk, majd ezek után könnyen és gyors módon felvenni a kapcsolatot az autószerelővel és továbbítani neki a kiosztott munkát. Ezeken túl titkárként lehetőségünk van munkákat lezárni, látni az éppen dolgozó autószerelőket vagy akár a már munkáját befejezett autószerelőt. Szerelőként lehetőségünk van bejelentkezni és látni a titkár által kiosztott munkát. Ezen felül néhány esetben a munkás is letudja zárni a munkát.

A webes alkalmazás fejlesztéséhez Blazorban szerzett legjobb programozási ismereteinket és az MsSql-ben szerzett tudásunkat használtuk fel.

- Magához az alkalmazáshoz Visual Studio 2022 programozási környezetet használtunk Radzen és c# nyelven.
- Az adatbázis tervezéshez és megvalósításához és ezen felül az adatok tárolásában az MsSql adatbázisrendszert alkalmaztuk.
- Api lekérésekhez a POSTMAN alkalmazást alkalmaztuk

A programban látható illusztrációkat, képeket az irodalomjegyzékben jelölt oldalról használtuk.

A Blazor a Microsoft által fejlesztett keretrendszer, amely lehetővé teszi a C# és .NET alapú webes alkalmazások fejlesztését. A Blazor segítségével JavaScript helyett C#-ban írhatunk interaktív frontend alkalmazásokat. Főbb előnyei közé tartozik, hogy .NET alapú üzleti alkalmazásokat könnyen létre lehet vele hozni, amely a mi estünkben egy nagyon nagy előny. Azért alkalmaztuk a blazor alkalmazást ugyanis mi egy c# alapú webes alkalmazást készítettünk.

A Radzen egy low-code fejlesztői eszköz, amelynek célja, hogy segítse a fejlesztőket modern, adatközpontú webalkalmazások gyors létrehozásában. Lényegében egy vizuális fejlesztőkörnyezet, ahol az alkalmazások különféle részeit (adatbázisok, üzleti logika, felhasználói felület, jogosultságok stb.) grafikus módon lehet megtervezni, a háttérben pedig a Radzen automatikusan legenerálja a hozzájuk tartozó forráskódot.

A Radzen erőssége abban rejlik, hogy leegyszerűsíti az üzleti alkalmazások fejlesztésének komplexitását, és jelentősen csökkenti a fejlesztési időt. Azért is használtuk a radzent ugyanis előnyeihez tartozik a könnyed fejlesztés, könnyű adatbázis-integráció, nyílt forrású, illetve vannak benne testre szabható generált kódok. A Radzen képes automatikusan leképezni az adatbázis struktúráját (pl. SQL Server, MySQL, PostgreSQL, Oracle stb.) és ezek alapján generálja az entitásmodelleket, szolgáltatásokat (Data Services), és az adminfelületet. A radzen automatikusan támogatja az adaptív megjelenítést különböző képernyőméretekre.

2. Hasonló webes alkalmazások

Mivel mi egy olyan webes alkalmazást készítettünk, amely kifejezetten egy cégre orientálódik ezért az olyan alkalmazások, amelyek megközelítik a mi alkalmazásunkat szinte nem is léteznek. Ugyanis a mi webes alkalmazásunk úgy készült, hogy egy megadott autószerelő cégnek minden igényét teljesíthessük. Ezen felül általánosságban az ilyen programok leginkább asztali alkalmazásként készülnek el míg a miénk egy webes alkalmazás. Ezen túl a mi alkalmazásunk sokkal gyorsabb kapcsolatot kínál, mint a többi asztali alkalmazásként elkönyvelt programok.

3. Felhasználói dokumentáció

A program célja az, hogy egy könnyen létrehozható kapcsolatot biztosítson a szerelők és titkárok között.

Lehetőségeink titkárként:

- Munka létrehozása
- Munka lezárása
- Adatok felvétele
- Bejelentkezés

Lehetőségeink szerelőként:

- Látni a folyamatban lévő munkát
- Látni a befejezett munkákat
- Egyes esetekben munka lezárása

Lehetőségeink adminként:

- Új titkárok hozzáadása, szerkesztése
- Új szerelők hozzáadása, szerkesztése
- Összes munkát látja
- Mindent tud szerkeszteni

3.1. Rendszer követelmények

A webes alkalmazásunk egy adatbázisban fut, ezért szükséges hozzá internet hozzáférés.

Minimális rendszerkövetelmény:

- Legalább 1,6GHz-es processzor
- Legalább 1GB szabad memória
- Legalább 10GB szabad tárhely
- Windows 7 vagy annál újabb operációs rendszer

Ajánlott rendszerkövetelmény:

- Legalább 2,5GHz-es processzor
- Legalább 2GB szabad memória
- Legalább 20GB szabad tárhely
- Windows 10 vagy annál újabb operációs rendszer

Ajánlott webes böngészők a futtatáshoz:

- Microsoft Edge
- Google Chrome
- Mozilla Firefox
- Opera
- Apple Safari

3.2. Belépés

Ahhoz, hogy egy felhasználó beléphessen az alkalmazásba először regisztrálni kell őt és utána már be is tud jelentkezni. Azt fontos megjegyezni regisztrálni új felhasználót csak is az adminisztrátor tud. Miután regisztrálva lett a felhasználó az adatai egy adatbázisban tárolódnak és az alapján tud belépni a felhasználó. Ha valamelyik adat helytelen vagy üres ebben az esetben a felhasználó nem tud belépni a fiókjába.

Az adminisztrátornak lehetősége van titkárokat, illetve szerelőket regisztrálni egy adatbázisban. A regisztrációhoz szüksége van egy névre, amely a későbbiekben a felhasználó név lesz, ezek mellett email cím, illetve egy jelszó. Ezek mellett az adminisztrátor a későbbiekben megtudja változtatni a felhasználó számára szükséges bejelentkezési adatokat is. Azonban felhasználót nem lehet törölni.

A későbbiekben a bejelentkezéshez szükséges egy megadott felhasználónév-kulcs páros megadása, ezek beírása után a „Bejelentkezés” gombra kattintással tud belépni a felhasználó. Miután belépett a felhasználó már láthatja is a számára fontos szolgáltatásokat.

3.3. Főoldal és tartalom

Az oldalunknak kettő fő része van. Egyszer van a titkár szemszögéből látott oldal, illetve a másik a szerelő szemszögéből látott oldal. Ezek mellett van egy oldalrész amely csak is az admin számára lett kialakítva és oda csak is az admin tud belépni.

A titkár főoldalán látható egy navigációs sáv, amely három helyre tudja tovább vinni a felhasználót. Egyik lehetőség az, ahol a titkár látja az összes létrehozott munkalapot, míg a másik nézetben a szerelőket látja, hogy melyik szerelő szabad es melyik szerelőnek van munkája, annak érdekében, hogy kit tud felvenni egy új munkára. A harmadik pedig a profil nézet, ahol a titkár tud jelszót változtatni.

A szerelő főoldalán egy navigációs sáv látható, amely kettő részre tudja tovább vinni a szerelőt. Az egyik ilyen rész a munkafolyamat részleg, ahol látja a számára kiosztott munkát és meg tudja nyitni őket hozzá adni folyamatokat. A lezártakat is láthatja, de nem tudja megnyitni.

Az admin oldalán az admin tud új felhasználókat hozzáadni, felhasználói adatokat módosítani, viszont törölni felhasználót nem tud. Ezek mellett az adminisztrátornak hozzáférése van az adatbázishoz, ezek mellett az adminisztrátor láthatja az összes kiosztott munkát, akár a folyamatban lévő munkákat akár a már lezárt munkákat. Fontos megjegyezni, hogy az adminisztrátor az adatbázisból adatokat csak különleges esetekben törölhet ki, jogi problémák elkerülésének az érdekében.

3.4. Funkciók ismertetése

3.4.1. Munkafolyamat hozzáadása

Munkafolyamatot csakis a titkár tud létrehozni. Itt lehet a különböző munkákat hozzáadni. A munkafolyamat hozzáadásához számos dolgot kell hozzá adni ilyen például a jármű típusa, benzin vagy dízel vagy, jármű rendszáma, ügyfél adatai ide tartozik telefonszám, email, név. Fontos, hogy ebben a szakaszban a titkárnak pontosan kell megadni minden egyes adatot ugyanis, ha nem ez történik a későbbiekben félreértés következhetne be.

Miután a titkár mind beírta a szükséges adatokat utána egy save gomb segítségével tudja hozzáadni a munkát és láthatóvá tenni a szerelő számára. Miután elküldte a szerelőnek az adatokat már tud is létrehozni újabb munkákat és újra kiosztani egy szerelőnek.

3.4.2. Munkafolyamat megtekintése

A munkafolyamat megtekintésére mind a három fél képes, de ez a szerelő számára a fontosabb ugyanis itt látja, hogy mennyi, illetve milyen munkákat osztottak ki neki. Egyszerre több munkája is lehet, illetve egyesesetekben ő is tud folyamatot hozzá adni. Viszont munkát egyáltalán nem tud törölni. Ahhoz, hogy ez az oldal látható legyen számára a navigációs részben a munkafolyamat feliratra kell nyomnia és már látja is. A titkár számára annyit jelent a munkafolyamat megtekintése, amikor a szerelő végez, ő elküldi a titkárnak és a titkár ott átnézheti. hogy minden a rendes módon történt e. Ezek mellett az adminisztrátor is látja az összes munkafolyamatot pontosan azért ugyanis ő felel minden egyes problémáért, ami esetleg felüti a fejét.

3.4.3. Munkafolyamat lezárása

Ez csak a szerelő számára látható, akkor kell lezárnia a munkafolyamatot amikor teljesen kész van a kiosztott munkájával. Ahhoz, hogy egy folyamat lezárt legyen rá kell nyomnia a munkafolyamat lezárása gombra. Amikor a szerelő rá nyom a munkafolyamat lezárására onnantól kezdve elküldte a titkár számára, aki láthatja, viszont már egyikük sem tudja megváltoztatni az adatokat a lezárt munkafolyamatban.

3.4.4. Lezárt munkafolyamat megtekintése

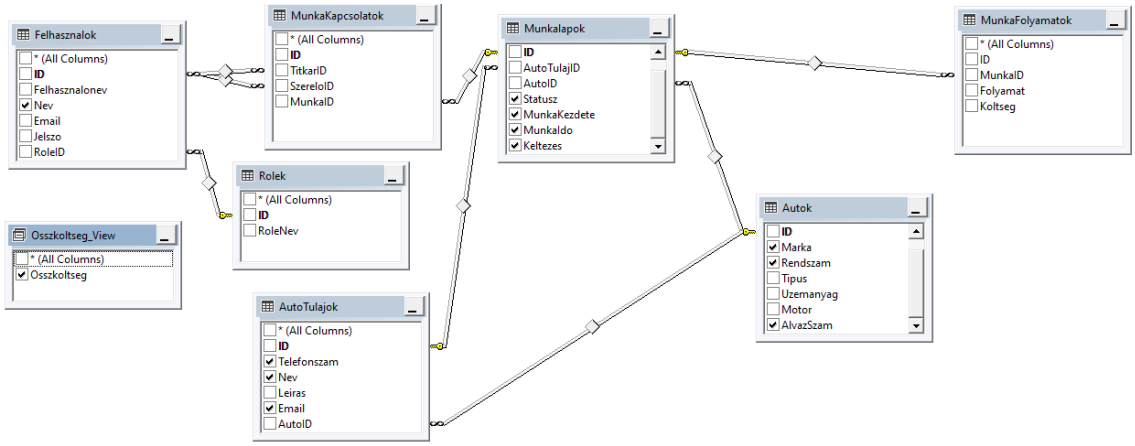
Ez a lehetőség mindhárom fél számára lehetséges, a titkár meg is tudja nyitni a lezárt munkafolyamatot, azért, hogy ellenőrizhesse, hogy a szerelő mindent jól csinált e és teljesített e mindent. Ezzel ellentétben a szerelő csak látja a lezárt munkafolyamatot, de megnyitni nem tudja ezzel elkerülve a csalás esélyét. Ezek mellett az adminisztrátor is látja a lezárt munkafolyamatokat.

3.4.5. Admin lehetőségek

Az admin számára egy teljesen külön oldal látható. Maga az admin látja az összes folyamatban lévő munkát és az összes lezárt munkát. Az admin tud törölni munkát, illetve tud új felhasználókat hozzáadni. Törölni viszont nem tud pontosan a félreértések elkerülése miatt. Ezen felül az admin hozzá fér az adatbázishoz, ahonnan szintén tud különböző adatokat törölni. Látja az adatbázishoz hozzáadott összes adatot, legyen szó akár különböző munkákról, vagy akár különböző felhasználókról (titkárok, szerelők).

4. Fejlesztői dokumentáció

4.1. Az adatbázis táblái



1. ábra az adatbázis struktúrájáról

Az alkalmazáshoz tartozó adatok tárolására Szakdolgozat nevű adatbázis nyújt lehetőséget. Az adatbázisnak a lokális szerveren kell futnia jelszó nélkül. Jelenleg az adatbázisunk 7 darab táblából és közöttük 8 kapcsolatból áll. Továbbá az adatbázisunk tartalmaz alapból legenerált adatokat, amelyek szerepet játszanak az alkalmazás tesztelésében.

4.1.1. Munkafolyamatok tábla

- A munkafolyamatok tárolására szolgáló tábla.

Mezőnév	**Típus**	**Megjegyzés**
ID	uniqueidentifier	Egyedi azonosító
MunkaID	uniqueidentifier	Egyedi azonosító
Folyamat	varchar	Folyamat leírása
Koltseg	bigint	Költség leírása

4.1.2. Felhasznalok tábla

- A felhasználók tárolására szolgáló tábla.

Mezőnév	**Típus**	**Megjegyzés**
ID	uniqueidentifier	Egyedi azonosító
Felhasznalonev	nvarchar	Felhasználónév leírása
Nev	nvarchar	Név leírása
Email	nvarchar	Email leírása
Jelszo	nvarchar	Jelszo leírása
RoleID	uniqueidentifier	Egyedi azonosító

4.1.3. Role tábla

- Ez a tábla segít elkülöníteni a jogosultságokat a felhasználók között.

Mezőnév	**Típus**	**Megjegyzés**
ID	uniqueidentifier	Egyedi azonosító
RoleNev	varchar	Rolenév leírása

4.1.4. Autok tábla

- Az autók tárolására szolgáló tábla

Mezőnév	**Típus**	**Megjegyzés**
ID	uniqueidentifier	Egyedi azonosító
Marka	nvarchar	Márka leírása
Rendszam	varchar	Rendszám leírása
Tipus	nvarchar	Típus leírása
Uzemanyag	nvarchar	Üzemanyag leírása
Motor	nvarchar	Motor típusa
AlvazSzam	nvarchar	Alvázszám leírása

4.1.5. Autotulajok tábla

- Az autotulajok tárolására szolgáló tábla

Mezőnév	**Típus**	**Megjegyzés**
ID	uniqueidentifier	Egyedi azonosító
Telefonszam	varchar	Telefonszám leírása
Nev	nvarchar	Név leírása
Leiras	nvarchar	Leírás leírása
Email	nvarchar	Email leírása
AutoID	uniqueidentifier	Egyedi érték

4.1.6. MunkaKapcsolatok tábla

- A munkakapcsolatok tárolására szolgáló tábla

Mezőnév	**Típus**	**Megjegyzés**
ID	uniqueidentifier	Egyedi azonosító
TitkarID	uniqueidentifier	Egyedi azonosító
SzereloID	uniqueidentifier	Egyedi azonosító
MunkaID	uniqueidentifier	Egyedi azonosító

4.1.7. Munkalapok tábla

- A munkalapok tárolására szolgáló tábla

Mezőnév	**Típus**	**Megjegyzés**
ID	uniqueidentifier	Egyedi azonosító
AutoTulajID	uniqueidentifier	Egyedi azonosító
AutoID	uniqueidentifier	Egyedi azonosító
Statusz	int	Státusz leírása
MunkaKezdete	datetime	Munkakezdet leírása
MunkaIdo	int	Munkaidő típusa
Keltezes	datetime	Keltezés leírása

4.1.8. Az összes kapcsolat az adatbázisban

Felhasznalok tábla kapcsolatai

- Felhasznalok.id → Munkakapcsolatok.TitkarID
- Felhasznalok.id → Munkakapcsolatok.SzerelokID
- Felhasznalok.RoleID → Rolek.ID

Munkakapcsolatok tábla kapcsolatai

- Munkakapcsolatok.MunkaID → Munkalapok.ID

AutoTulajok tábla kapcsolatai

- AutoTulajok.ID → Munkalapok.AutoTulajID
- AutoTulajok.AutoID → Autok.ID

Munkalapok tábla kapcsolatai

- Munkalapok.ID → Munkafolyamatok.MunkaID
- Munkalapok.AutoID → Autok.ID

4.2. Componensek

A szoftverfejlesztésben a komponens egy újrafelhasználható kódblokk, amely általában egy bizonyos feladatot lát el, például egy gomb, egy űrlap vagy egy navigációs menü. A modern fejlesztési keretrendszerek, mint például a React, Angular vagy Vue.js, széles körben használják a komponens-alapú fejlesztést, ahol az alkalmazás felépítése kisebb, kezelhető egységekből, komponensekből történik.

Kód újrahasználhatóság: A komponensek önálló egységek, amelyeket könnyedén újrahasználhatsz más projektekben. Ezáltal csökkentheted a kód duplikációját, és gyorsabban fejleszthetsz.

Modularitás: A komponens-alapú megközelítés elősegíti a rendszer modularitását, ami azt jelenti, hogy a különböző részek külön-külön fejleszthetők, tesztelhetők és frissíthetők anélkül, hogy a rendszer más részeinek működését befolyásolnánk. Előny: Könnyebbé válik a kód karbantartása és frissítése, mivel a hibákat könnyebben lokalizálhatjuk és javíthatjuk.

Rugalmasabb fejlesztés: A komponensek lehetővé teszik, hogy a rendszert kisebb, kezelhetőbb darabokra bontsd. A különálló komponensek lehetővé teszik a párhuzamos munkát, mivel a csapatok egy-egy komponens fejlesztésére koncentrálhatnak.

Több fejlesztői csapat dolgozhat egyszerre: A komponensek lehetővé teszik, hogy különböző csapatok párhuzamosan dolgozzanak, mivel a munkájukat jól elkülöníthetik egymástól. Egyik csapat dolgozhat például az adatkezelésen, míg egy másik a felhasználói felület elemeinek fejlesztésén.

Jobb tesztelhetőség: A komponens-alapú fejlesztés lehetővé teszi a komponensek önálló tesztelését. Mivel minden komponens izolált egységként működik, könnyen elvégezhetők az egyes komponensek egységtesztjei. Előny: Könnyebb és gyorsabb tesztelés, jobb hibakeresés és megbízhatóság.

Összegzés: A komponens-alapú megközelítés az alkalmazások fejlesztésében hatékonyabbá teszi a munkát, könnyíti a karbantartást, gyorsabbá teszi a fejlesztési folyamatokat, és elősegíti a rendszer hosszú távú fenntarthatóságát. Az előnyök különösen a nagyobb, összetettebb rendszerek és alkalmazások fejlesztésénél válnak szembetűnővé.

Szakdolgozatunk során számos Radzen komponenst alkalmaztunk.

Ezek közül pár db:

„Radzen DataGrid”: Az egyik legfontosabb és leggyakrabban használt UI-komponens a Radzen keretrendszerben, mivel lehetővé teszi a táblázatos adatok hatékony megjelenítését és kezelését.

„Login Form”: Egy olyan űrlap, amely a felhasználó bejelentkezéséhez szükséges, és ahol meglehet adni a felhasználónevet és a jelszót.

„Button”: A Radzen Button segítségével felhasználói interakciókat hozhatunk létre, és a gombok különböző eseményeket válthatnak ki, például adatküldést, műveletek elindítását, vagy egyéb UI-frissítéseket.

„Link”: A Radzen Link lehetővé teszi, hogy hivatkozásokat hozzunk létre az alkalmazásunkban, amelyek más oldalra vagy külső URL-re vezetnek.

„Menu”: A Radzen Menu lehetővé teszi a többszintű menüstruktúrák létrehozását, tehát alkategóriák és legördülő menük segítségével könnyen kezelhetjük a nagyobb és összetettebb navigációkat.

„Table”: A Radzen Table lehetővé teszi, hogy különböző típusú adatokat, mint például objektumok, listák vagy akár adatbázisokban tárolt rekordok, táblázatos formában jelenítsük meg. A táblázat sorai és oszlopai dinamikusan beállíthatók, lehetőséget biztosítva az adatok testreszabására. Az oszlopok tartalma módosítható, például szövegekkel, számokkal, képekkel vagy egyéb komponensekkel.

„Icon”: A Radzen Icon különböző ikonokat kínál, amelyek könnyen testreszabhatók és használhatók az alkalmazások felhasználói felületén. Az ikonok lehetnek statikusak, vagy dinamikusan vezérelhetők események, stílusok, színek, és más paraméterek segítségével.

4.3. Github

A szakdolgozatunk elkészítése során használatba vettük a Github nevű alkalmazást, amely számos segítséget nyújtott nekünk és számos előnnyel rendelkezik.

A GitHub egy webalapú platform, amelyet a fejlesztők és programozók használnak kódbázisok tárolására, megosztására és együttműködésre. Az egyik legnépszerűbb eszköz a verziókezeléshez, és a Git nevű verziókezelő rendszerre épít. GitHub lehetővé teszi a kódok könnyű verziókövetését, együttműködést a csapatok számára, és sok más hasznos funkcióval rendelkezik a szoftverfejlesztési folyamatokban.

Ezen előnyök közül pár:

Verziókezelés: A GitHub alapja a Git verziókezelő rendszer, amely lehetővé teszi a kód folyamatos nyomon követését és visszatekintést a különböző verziók történetébe. Így könnyen nyomon követhetők a változtatások, és bármikor vissza lehet térni egy előző verzióhoz.

Kollaboráció: A GitHub remekül támogatja a csapatmunkát, mivel lehetővé teszi a több fejlesztő számára, hogy egyszerre dolgozzanak ugyanazon a projekten. A pull requestek és a branch-ek (ágak) segítségével a fejlesztők könnyen dolgozhatnak párhuzamosan, majd integrálhatják munkájukat.

Projektek kezelése: GitHub Project Boards és Issues rendszerei segítségével könnyedén nyomon követhetők a feladatok, hibák és új funkciók. Ez különösen hasznos nagyobb csapatok esetében, ahol sok feladatot kell koordinálni.

Biztonság és hozzáférés-vezérlés: GitHub lehetővé teszi, hogy különböző szintű hozzáféréseket állítsunk be a repositorykhoz (pl. adminisztrátor, olvasó, író). Ez segít a projekt védelmében és a biztonság fenntartásában.

Összegzés: A GitHub tehát egy kulcsfontosságú eszköz a szoftverfejlesztésben, amely lehetővé teszi a verziókezelést, a csapatmunkát, az open-source projektek támogatását és számos fejlesztési folyamat automatizálását. Ha programozó vagy fejlesztő vagy, a GitHub rendkívül hasznos eszköz lehet a munkádban!

4.4. Jelszó titkosítása

Alkalmazásunkban a jelszó titkosítása fontos szempont, ugyanis megakarjuk előzni a csalásokat, illetve az esetleges támadásokat. Webes alkalmazásunkban a jelszó titkosítása mind a titkárra, mind a szerelőre és mind az adminisztrátorra is kiterjed. A fejlesztői környezet, amellyel mi dolgozunk kettő fő módon tudja ellátni a jelszó titkosítási problémát. A többet használt és a többet ajánlott az a ASP.NET Identify míg a másik lehetőség az a kézi hasheelés. Mi az ASP.NET Identifyt alkalmaztuk ugyanis a kézi hasheléssel ellentétben ez automatikusan kezeli a magát a hashelést ezek mellett biztonságosabb, mint a kézi hashelés, illetve beépített felhasználókezelést és adatbázis-integrációt kínál számunkra.

4.4.1. Hogyan működik

Az ASP.NET Identity egy beépített felhasználókezelő rendszer, amely lehetővé teszi a felhasználók regisztrációját, bejelentkezését**,** szerepköreit és jogosultságkezelését. Támogatja az adatbázisba mentett jelszavak biztonságos hash-elését és egyéb hitelesítési megoldásokat (pl. Google, Facebook, JWT Tokenek).

Az ASP.NET Identity egy adatbázisban tárolja a felhasználókat és a hozzájuk tartozó információkat. Főbb összetevői:

- **UserManager**: A felhasználók kezelésére szolgál (pl. regisztráció, jelszóellenőrzés, bejelentkezés).
- **SignInManager:** A bejelentkezési műveleteket végzi (pl. cookie-k, kétlépcsős hitelesítés).
- **IdentityUser:** Az alapértelmezett felhasználói osztály, amely olyan adatokat tárol, mint az e-mail, jelszó, telefonszám stb.
- **IdentityRole:** Szerepköröket kezel (pl. "Admin", "User").
- **DbContext:** Az adatbáziskapcsolatot biztosítja, az Entity Framework Core segítségével.

Jelszó tárolás és titkosítás: Az ASP.NET Identity a bcrypt algoritmust használja a jelszavak titkosítására. A rendszer sózást és iterációkat alkalmaz, hogy megnehezítse a brute force támadásokat.

Előnyei az ASP.NET Identity-nek:

- Biztonságos jelszókezelés (bcrypt hashing, sózás, iterációk)
- Könnyen integrálható az Entity Framework Core-al
- Támogatja a kétlépcsős hitelesítést (2FA)
- Szerepkörök és jogosultságkezelés

Összegzés: Az ASP.NET Identity egy erőteljes és biztonságos felhasználókezelő rendszer, amely lehetővé teszi:

- A felhasználók kezelését (regisztráció, bejelentkezés, jelszómódosítás)
- Szerepkörök és jogosultságok kezelését
- Több hitelesítési módszert

4.5. Autentikáció

Az autentikáció egy olyan folyamat, amely során egy rendszer vagy alkalmazás megerősíti, hogy a felhasználó valóban az, akinek mondja magát. Az autentikáció célja a felhasználó személyazonosságának ellenőrzése és biztosítása, hogy a rendszerhez való hozzáférés csak a jogosult személyek számára legyen engedélyezve. Ez a folyamat kulcsfontosságú a biztonságos alkalmazások és rendszerek működésében.

Ez a folyamat több lépésből áll:

- **Felhasználónév azonosítása:** Ez az a folyamat, amely során, amikor a felhasználó bejelentkezik, a háttérben működő adatbázisban a szerver ellenőrzi, hogy a felhasználó által megadott felhasználónév létezik e és létre van e hozva.
- **Jelszó ellenőrzése:** Miután a szerver ellenőrizte, hogy létezik e a megadott felhasználónév utána ellenőrzi a felhasználó által beírt jelszót a többi tárolt jelszóval. Ha ez sikeres akkor megtörténhet a bejelentkezés.
- **Süti azonosítók:** Segítenek emlékezni a weboldalnak a felhasználó preferenciáiról ezek mellett a bejelentkezési adatokról és még sok más fontos adatról. Fő célja. hogy javítsa a felhasználói élményeket.

Az autentikáció biztonságára pédák:

- **Erős jelszavak**: A gyenge jelszavak, mint a "123456" vagy a "password", könnyen kitalálhatóak és veszélyeztethetik a fiók biztonságát. Az erős jelszavak hosszúak, vegyes karaktereket tartalmaznak, és elkerülik a könnyen kitalálható szavakat.

- **Többfaktoros autentikáció:** A többfaktoros autentikáció jelentősen növeli a biztonságot, mivel egy támadó számára nehezebb megszerezni mindkét autentikációs tényezőt (például egy jelszót és egy telefonon keresztül küldött SMS kódot).

Összegzés: Az autentikáció tehát egy alapvető biztonsági mechanizmus, amely biztosítja, hogy a felhasználók csak a megfelelő hozzáférési jogosultságokkal rendelkező személyek számára tudjanak belépni egy rendszerbe. A különböző autentikációs típusok (pl. jelszavak, biometrikus azonosítás, MFA) különböző szintű védelmet nyújtanak, és az alkalmazott módszerektől függően biztosíthatják a személyes és céges adatok biztonságát.

Ezek mellett vannak még lépései az autentikációnak, de számunkra az előbbiekben felsoroltak a legfontosabb elemek.

4.6. Tesztdokumentáció

A fejlesztés során nem csak a kódolásra fektettünk nagy hangsúlyt, hanem ezzel páhuzamban a folyamatos tesztelésekre is ugyan akkora hangsúlyt fektettünk. Kódolással párhuzamban hajtottuk végre a tesztelést tehát ha valami újat készítettünk azt utána le is teszteltük, hogy tudjunk tovább haladni. A program elkészülte után a csapat összes tagja többszöri alkalommal tesztelte a programot annak az érdekében, hogy minden 100%-os legyen és hogy egy tökéletes webes alkalmazást adhassunk át.

A tesztelés folyamatához mindig elindítottuk a webes alkalmazást és mindig az új kódhoz fűződő új elemeket adtunk hozzá és győződtünk meg arról, hogy rendben van e minden.

5. Weblap designja

A weblap kinézete eléggé sok gondolkozást vett magához ugyanis a mi oldalunknak nem az a fő célja, hogy jól nézzen ki, hanem hogy könnyen alkalmazható legyen és kényelmes legyen a használata. De ezek mellett arra is gondolnunk kellett, hogy a színek és a maga a logo is teljes mértékben tükrözze a szervízt. Ezért is döntött úgy a csapat, hogy nem a kinézet lesz az elsődleges, hanem magának a programnak és magának a már kész és befejezett webes alkalmazásnak a működése. Ezért is rendelkezik az alkalmazásunk minimális design-val.

De azt a minimális kinézetet, amit létrehoztunk a program fejlesztésével folyamatosan fejlesztettük.

A fejlesztés során bár nem használtunk fel sok képet sem sok ikont, de amit alkalmaztunk az a Canva nevű alkalmazás segítségével értük el. Gondolkoztunk képszerkesztő alkalmazás használatára, de számunkra elég a minimális design-val rendelkező képek, illetve ikonok, így arra jutottunk, hogy sok értelme nem lenne ilyen alkalmazást használni. A logo létrehozására több logo készítő weboldal is tervben volt, de a kiválasztott oldal a Canva lett pontosan azért ugyanis könnyű felhasználói környezettel rendelkezik és számos olyan képet fel lehet használni, aminek nincsenek jogi problémái.

Az alkalmazás létrehozásában csak is olyan képeket, ikonokat alkalmaztunk, amelyek teljes mértékben jogtiszták és nem ütköznek jogi problémába.

6. Összefoglalás

6.1. A szakdolgozat fő célja

A szakdolgozatunkkal a fő célunk az volt, hogy létrehozzunk egy olyan webes alkalmazást, amely sokkal rugalmasabbá, egyszerűvé, és dinamikussá tegye egy autószerelő üzemben dolgozó titkárok és szerelők közti kapcsolatot. A projekt fő célja a könnyű adatátvitel a felek között.

6.2. Megvalósítása

Ahogyan projektünk fejlődött egyre jobban és jobban számos olyan probléma lépett fel, amit eddig nem ismertünk, de hála a kitartó csapatnak minden egyes problémát megtudtunk oldani és felülkeredtünk rajtuk. Az oldal megtervezésétől kezdve a különböző funkciók megvalósításáig haladtunk, minden új résznél más és más ismereteket kellett alkalmaznunk.

Az oldal kinézetével nem sokat foglalkoztunk ugyanis mi egy egyszerűen használható alkalmazást szerettünk volna létrehozni, ezáltal nem a kinézet volt a fő cél, hanem a funkcionalitás. A kinézet megtervezéséhez a Canva nevő weboldalt alkalmaztuk, amely segítségünkre volt a struktúrák kialakításában is. Az adatbázisunkat már a legelején létrehoztuk és ahogy haladtunk előre, ha kellett bővítettük.

A főoldal létrehozása során nem gondoltunk sokra ugyanis átlátható oldalt szerettünk volna ezért is döntöttünk úgy, hogy a főoldal kettő részből álljon. Külön a szerelők számára és külön a titkárok számára. Radzen segítségével hoztuk létre az oldalakat, amelyek rugalmasak és könnyen kezelhetőek lettek.

A bejelentkezési felület egy teljesen alap felület lett, ahol a felhasználónak megkell adnia a

felhasználónevét és a jelszavát.

A homepage-ek létrehozása nem okozott nagyobb problémát ugyanis ezek is az egyszerűségre törekedtek leginkább ezáltal semmi bonyodalom nem történt.

Az egyik legnehezebb rész az az új munka létrehozása és annak a láthatósága volt. Ezt úgy kellett végrehajtanunk, hogy amit a titkár hozzáad munkát azt a kiosztott szerelő már láthassa is viszont csak ritka esetben tudja módosítani. Az elakadásokat különböző oldalak segítségével oldottuk meg, ilyen például a StackOwerflow. Ennél a résznél már számított a kinézet ugyanis azt akartuk, hogy a felek rendesen és átláthatóan nézhessék át a számukra kiosztott munkát.

A titkár oldalon a fő feladatunk az volt, hogy létre tudjon új munkát hozni és ezt a létrehozott munkát egy oldalon lévő gomb segítségével továbbíthassa a szerelő számára.

A szerelő oldalának legfontosabb eleme az a rész volt, ahol látja a munkáit, amelyek lehettek folyamatban lévő munkák, amiket megtud nyitni és akár lehetnek már lezárt munkák, amiket viszont már nem tud megnyitni a csalás elkerülésének érdekében.

Az admin oldala az az oldal, amihez csak is az admin fér hozzá. Ennek a fő célja, hogy lássa az adatbázist, illetve, hogy létre tudjon hozni újabb szerelőket, illetve újabb titkárokat.

7. Fejlesztési lehetőségek

7.1. További lehetőségek, lapok hozzáadása

- **Beléptetés/Kiléptetés:** Amikor a szerelő elkezdi a munkaszakaszát rá nyom, hogy elkezdte a munkát, és amikor végzett pedig rá nyom a kiléptetésre, így elkerülve az órabeli eltolódást
- **Újabb táblázatok:** Az esetleges olyan táblázatok, ahol nincsen minden adat annak a pótlása, illetve helyettesítése
- **Adatbázis fejlesztése:** A jövőben, ha új adatokat szeretne a moderátor akkor lehetősége lesz új adatokat felvenni

7.2. Az elkezdett munka automatikus mentése

Az alap megoldás egy „mentés” gombbal menti el a megadott adatokat. Ez helyett lehetne, hogy az alkalmazás automatikusan menti el az adatokat attól függetlenül mennyi adatot vittünk fel. Tehát ha a titkár félbe hagyja valami miatt az sem baj mert elmentené automatikusan.

7.3. Profilbeállítások

7.3.1. Jelszó módosítása

A jelszó módosítására csak is az admin jogosult és csak is ő tudja módosítani vagy hozzá adni. Ez helyett lehetne az, hogy maga a titkár illetve maga a szerelő tudja módosítani a saját jelszavát a bejelentkezés oldalán.

8. Irodalomjegyzék

https://chatgpt.com/

https://stackoverflow.com/questions

https://www.figma.com/

https://github.com/

https://blazor.radzen.com/login?theme=material3

https://blazor.radzen.com/datagrid?theme=material3

https://blazor.radzen.com/button?theme=material3

https://blazor.radzen.com/?theme=material3

https://learn.microsoft.com/en-us/aspnet/core/blazor/tutorials/movie-database-app/part-4?view=aspnetcore-9.0&pivots=vs

https://learn.microsoft.com/en-us/aspnet/core/blazor/blazor-ef-core?view=aspnetcore-8.0

https://www.radzen.com/blazor-studio/documentation/databases/

https://stackoverflow.com/questions/63996294/how-to-connect-blazor-webassembly-to-database

https://learn.microsoft.com/hu-hu/shows/on-dotnet/on-dotnet-live-radzenblazor-a-free-and-open-source-component-library

https://visualstudio.microsoft.com/vs/

9. Mellékletek

9.1. Példa a felhasználó kiiratására

```
private List<Role> szerelokListaja;
    private Role selectedSzerelo;

    protected override async Task OnInitializedAsync()
    {
        szerelokListaja = await Role_Service.GetRolek();

        if (szerelokListaja.Any())
        {
            selectedSzerelo = szerelokListaja.First();

        }

    }

    [Inject]
    public NotificationService notificationService { get; set; }

    Felhasznalo felhasznalo = new Felhasznalo();
    Role role = new Role();

    void ShowNotification(NotificationMessage message)
    {
        notificationService.Notify(message);
    }

    private async Task OnSubmit()
    {
        if (string.IsNullOrWhiteSpace(felhasznalo.Felhasznalonev))
        {
            ShowNotification(new NotificationMessage
                {
                    Severity = NotificationSeverity.Error,
                    Summary = "Hiba",
                    Detail = "A felhasználónév megadása kötelező!",
                    Duration = 3000,

                });
            return;
        }
```

```
else if (string.IsNullOrWhiteSpace(felhasznalo.Nev))
        {
            ShowNotification(new NotificationMessage
                {
                    Severity = NotificationSeverity.Error,
                    Summary = "Hiba",
                    Detail = "A név megadása kötelező!",
                    Duration = 3000,

                });
            return;
        }

        else if (string.IsNullOrWhiteSpace(felhasznalo.Email))
        {
            ShowNotification(new NotificationMessage
                {
                    Severity = NotificationSeverity.Error,
                    Summary = "Hiba",
                    Detail = "Az email megadása kötelező!",
                    Duration = 3000,

                });
            return;
        }

        else if (!felhasznalo.Email.Contains("@")
|| !felhasznalo.Email.Contains("."))
        {
            ShowNotification(new NotificationMessage
                {
                    Severity = NotificationSeverity.Error,
                    Summary = "Hiba",
                    Detail = "Érvénytelen email cím formátum!",
                    Duration = 3000 });
```