



EÖTVÖS LORÁND TUDOMÁNYEGYETEM

INFORMATIKAI KAR

PROGRAMOZÁSI NYELVEK ÉS FORDÍTÓPROGRAMOK

TANSZÉK

Parkolóház foglalórendszer

Témavezető:

Fekete Anett

PhD hallgató

Szerző:

Csonka László

programtervező informatikus BSc

Budapest, 2023

EÖTVÖS LORÁND TUDOMÁNYEGYETEM

INFORMATIKAI KAR

SZAKDOLGOZAT TÉMABEJELENTŐ

Hallgató adatai:

Név: Csonka László

Neptun kód: H6XS3I

Képzési adatak:

Szak: programtervező informatikus, alapképzés (BA/BSc/BProf)

Tagozat : Nappali

Belső témavezetővel rendelkezem

Témavezető neve: Fekete Anett

munkahelyének neve, tanszéke: ELTE IK, Programozási nyelvek és Fordítóprogramok Tanszék

munkahelyének címe: 1117, Budapest, Pázmány Péter sétány 1/C.

beosztás és iskolai végzettsége: PhD hallgató MSc

A szakdolgozat címe: Parkolóház foglalórendszer

A szakdolgozat téma:

(A témavezetővel konzultálva adja meg 1/2 - 1 oldal terjedelemben szakdolgozat témájának leírását)

A mai rohanó világban, alig jut időnk valamire, tehát minden töreksünk arra, hogy amivel lehet a lehető legtöbb időt megspóroljuk.

Reggelente a legtöbb dolgozó szenvéd attól a problémától, hogy nagyon nehezen talál parkolót még a saját munkahelyén is, ami azért valljuk be

nagyon sok időt is igénybe vehet, ha a szerencse nem a mi oldalunkra áll. Ennek a webalkalmazásnak pontosan az lenne a célja, hogy ezt a problémát megoldja nekünk,

ugyanis lehetőséget biztosít a felhasználók számára, hogy a következő munkanapra lefoglalják parkolóhelyüket.

A foglalást egy nagyon egyszerű dizájnos felületen tehetik meg, ahol még a parkoló típusát is kitudják választani a felhasználók igényeik szerint pl.: elektromos, elsőbbségi.

Az alkalmazás használatához szükség van egy felhasználónévre és jelszóra, amelyet a cég admin fog kiosztani nekünk, ugyanis nem csak a felhasználóknak szükséges az azonosítás, hanem a cégeknek is.

A cégek minden parkolóházukat felvihetik a rendszerbe nagyon könnyen, amelyet akár a későbbiekben bővíthetnek vagy szerkeszthetnek is.

Az alkalmazás nem csak egy vagy több cég számára használható, hanem általános célú is.

A webalkalmazás arra törekszik majd, hogy a fent említett problémát minél rugalmasabban kezelje.

A rendszer C#, MSSQL és Angular technológiákkal fog megvalósulni.

Budapest, 2022. 10. 27.

Tartalomjegyzék

1. Bevezetés	4
2. Felhasználói dokumentáció	5
2.1. Az alkalmazás rövid ismertetése	5
2.2. Rendszer követelmények	5
2.2.1. Windows	5
2.2.2. macOS	6
2.2.3. Linux	6
2.3. Telepítés	6
2.4. Az alkalmazás felületei	6
2.4.1. Bejelentkezási felület	7
2.4.2. Regisztrációs felület	8
2.4.3. Jelszóváltoztatási felület	9
2.4.4. Home felület	10
2.4.5. Companies felület	11
2.4.6. Users felület	12
2.4.7. Parkinghouses felület	12
2.4.8. Slots felület	13
2.4.9. Reserve felület	14
2.4.10. My reservations felület	14
2.5. Az alkalmazás használata	15
2.5.1. Bejelentkezés	15
2.5.2. Regisztráció	15
2.5.3. Cégek, felhasználók, parkolóházak, szintek aktiválása és deaktiválása	15
2.5.4. Cégek, felhasználók, parkolóházak, szintek eltávolítása	16
2.5.5. Felhasználók felvétele	17

2.5.6. Parkolóházak felvétele	17
2.5.7. Szintek felvétele	18
2.5.8. Parkolóhelyek felvétele	19
2.5.9. Parkolóhelyek személyre szabása	20
2.5.10. Foglalás menete	22
3. Fejlesztői dokumentáció	24
3.0.1. Tervezés és specifikáció	24
3.1. Felhasználói felület	26
3.2. Fejlesztői környezet konfigurálása	28
3.2.1. Felhasználói felület fejlesztői környezet konfigurálása	28
3.2.2. API végpontok fejlesztői környezet konfigurálása	29
3.3. Alkalmazott technológiák	29
3.3.1. Angular	29
3.3.2. Visual Studio Code	29
3.3.3. Visual Studio Community 2022	30
3.3.4. Git, GitHub és GitHub Desktop	30
3.3.5. MSSQL	30
3.3.6. Draw.io	30
3.4. Felhasznált keretrendszerek	30
3.4.1. Entity Framework	30
3.4.2. Identity Framework	30
3.4.3. Bootstrap	31
3.5. Alkalmazás megvalósítása	31
3.5.1. Adatbázis séma	32
3.5.2. API végpontok osztálydiagramja	34
3.6. Tesztelési terv	36
3.6.1. Felhasználói felület tesztelése	36
3.6.2. API végpontok tesztelése	38
4. Összegzés	40
4.0.1. Továbbfejlesztési lehetőségek	40
Irodalomjegyzék	41

TARTALOMJEGYZÉK

Ábrajegyzék	43
Táblázatjegyzék	45

1. fejezet

Bevezetés

A mai rohanó világban, alig jut idő valamire, így az ember minden próbál arra törekedni, hogy amivel csak lehet, a lehető legtöbb időt megspórolja. Reggelente a legtöbb dolgozó szenvéd azzal a problémával, hogy nagyon nehezen talál parkolóhelyet még a saját munkahelyén is, ami valljuk be, nagyon sok időt is igénybe vehet, ha a szerencse nem az Ő oldalukon áll. Vannak emberek, akik korábban kelnek fel csak azért, hogy legyen hol parkolniuk, de van, hogy még így sem sikerül egyhamar parkolót találniuk. Az időt, amit szabad parkolóhely keresgélésével tölt a dolgozó nap mint nap, hasznosabban is tölthetné azt a tizenöt vagy húsz percet, talán a reggeli sietségben elfelejtett kávét is elfogyaszthatná, s már könnyebben és frissebben indulna neki a napnak.

Célom, egy olyan webalkalmazás készítése, ami pontosan ezt a problémát oldja meg, ugyanis lehetőséget biztosít a felhasználók számára, hogy a következő munkanapra lefoglalják a parkolóhelyüket. A foglalást egy letisztult és könnyen kezelhető felületen tehetik meg, ahol még a parkoló típusát is a felhasználók döntik el. Saját igényeik szerint választhatnak például elektromos, elsőbbségi vagy akár rokkant parkolóhelyek közül is. Ezzel sok időt megspórolhat az ember és zökkenőmentessé válik a reggeli parkolás is a zsúfolt munkahelyeken. Mivel minden napra rákövetkezendő napra lehet foglalni, így biztos, hogy mindenki egyenlő esélyekkel foglalhat igényeinek megfelelően parkolóhelyet.

2. fejezet

Felhasználói dokumentáció

2.1. Az alkalmazás rövid ismertetése

Az webalkalmazás egy egyszerű megoldást biztosít a pakolóhelyek keresése okozta kellemetlenségekre. Lehetőséget ad a felhasználók, parkolóházak, szintek és parkolóhelyek kezelésére.

A szoftver különböző jogosultságokhoz rendelt felületeket biztosít a zökkenőmentes parkolóhely foglalásra és azok kezelésére.

A következő jogosultságokkal találkozhat a felhasználó az alkalmazásban:

- **SystemAdmin** - A regisztrált cégek kezelésért felelős;
- **CompanyAdmin** - Az adott cégért felelős;
- **Boss/Employee** - Az adott cég dolgozója.

2.2. Rendszer követelmények

2.2.1. Windows

- OS: Windows 10 64-bit vagy későbbi
- Processzor: 64-bit processzor
- Memória: 4 GB
- BIOS-level hardver virtualizáció támogatás engedélyezése
- Hyper-V Windows szolgáltatások engedélyezése

2.2.2. macOS

- OS: macOS 10.15 vagy újabb (Monterey, Big Sur, Catalina)
- Memória: 4 GB
- Mac hardverje 2010-es vagy annál újabb
- Processzor: Intel

2.2.3. Linux

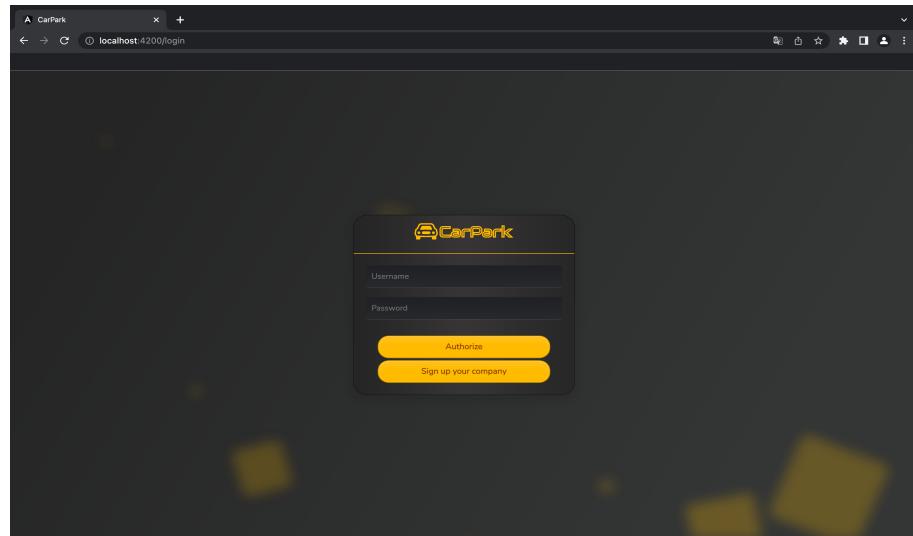
- Processzor: 64-bit processzor
- Memória: 4 GB
- KVM virtualizáció támogatás
- QEMU 5.2 vagy újabb

2.3. Telepítés

Feltételezve, hogy a program forráskódja a számítógépünkön található, a program Docker[1] környezetben fut, amely lehetővé teszi bármelyik platformon a használatát. A Docker telepítése után a program elérési könyvtárában adjuk ki a `docker-compose build` parancsot, amely a `docker-compose.yml` fájl alapján lefordítja a `Dockerfile`-kat. A parancs lefutása után adjuk ki a `docker-compose up` parancsot, amely elindítja a szükséges szolgáltatásokat és csatolja ehhez a konténeket.

2.4. Az alkalmazás felületei

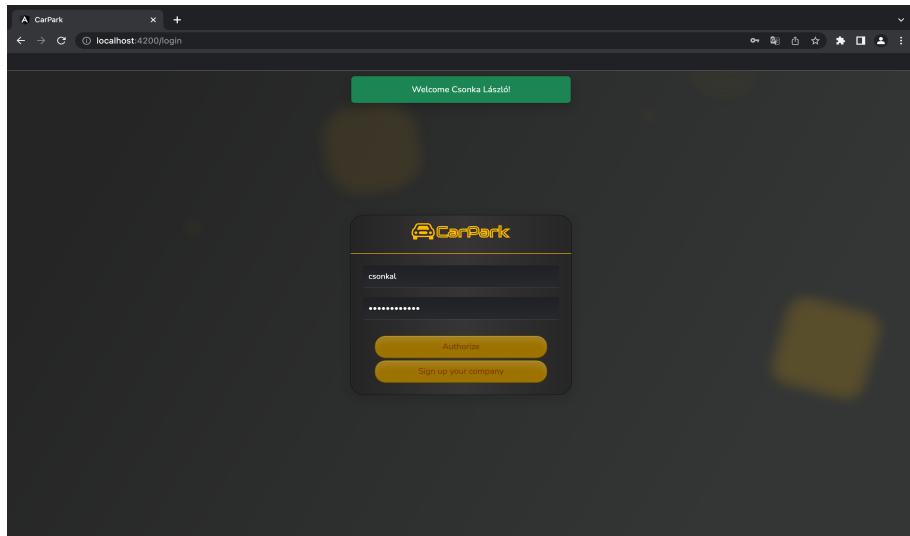
Az alkalmazás a sikeres telepítés után a `http://localhost:4200` -as felületen érhető el bármelyik böngészőből, ahol a betöltés után a bejelentkezési felület fogadja a felhasználót.



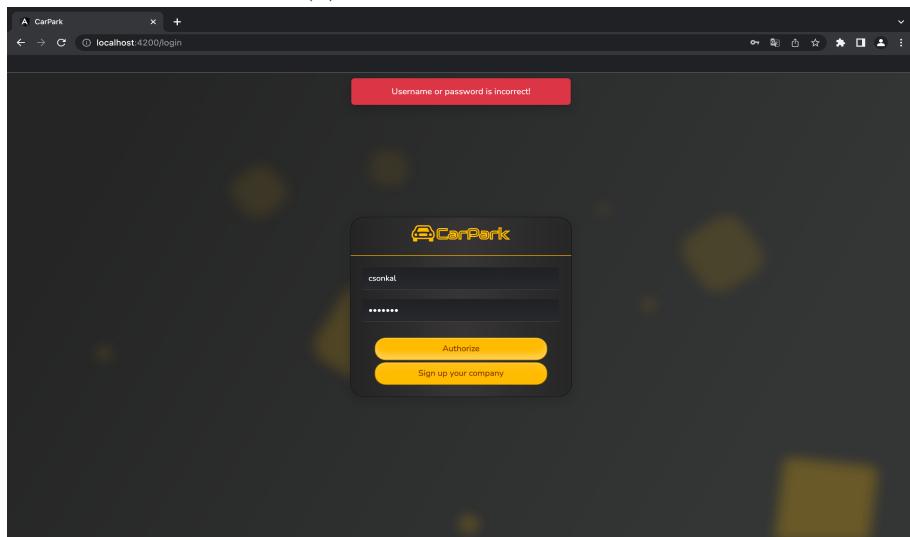
2.1. ábra. Bejelentkezési felület

2.4.1. Bejelentkezési felület

A bejelentkezési felületen a felhasználó végrehajthatja a bejelentkezést, amelynek sikerességről az oldal rögtön tájékoztatja. Lehetősége van a felhasználónak a regisztrációs felületre navigálni.



(a) Sikeres bejelentkezés



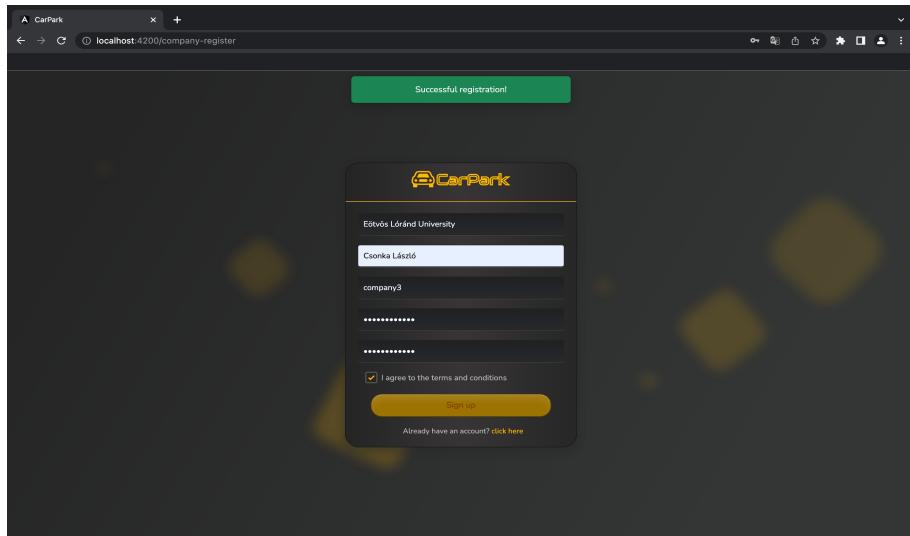
(b) Sikertelen bejelentkezés

2.2. ábra. Bejelentkezási felület

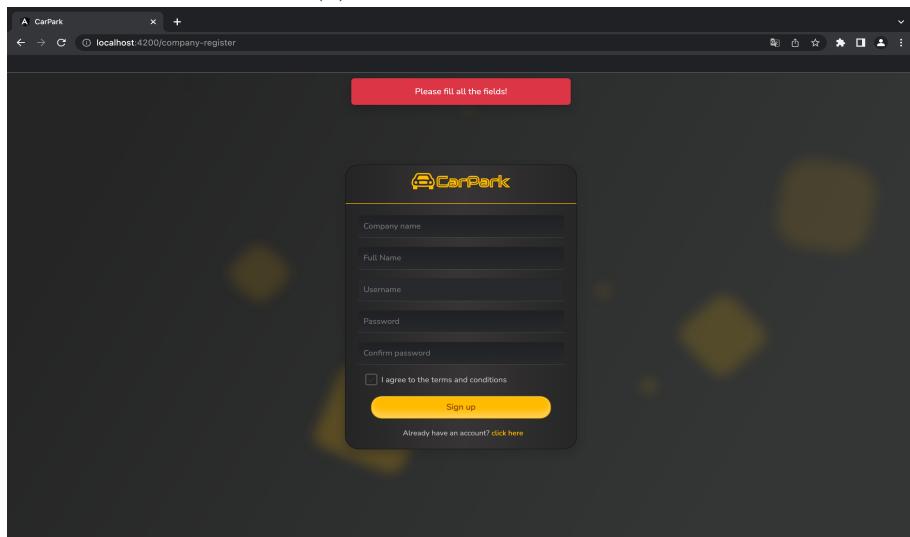
2.4.2. Regisztrációs felület

A regisztrációs felületen a felhasználó regisztrálhatja saját cégett. A regisztráció folyamán az oldal folyamatosan tájékoztatja arról, hogy minden adatmezőt megfelelően töltött-e ki. Amennyiben ezt sikerült végrehajtania, az oldal sikeres visszajelzést küld számára és automatikusan átnavigálja a bejelentkezási felületre.

2. Felhasználói dokumentáció



(a) Sikeres regisztráció

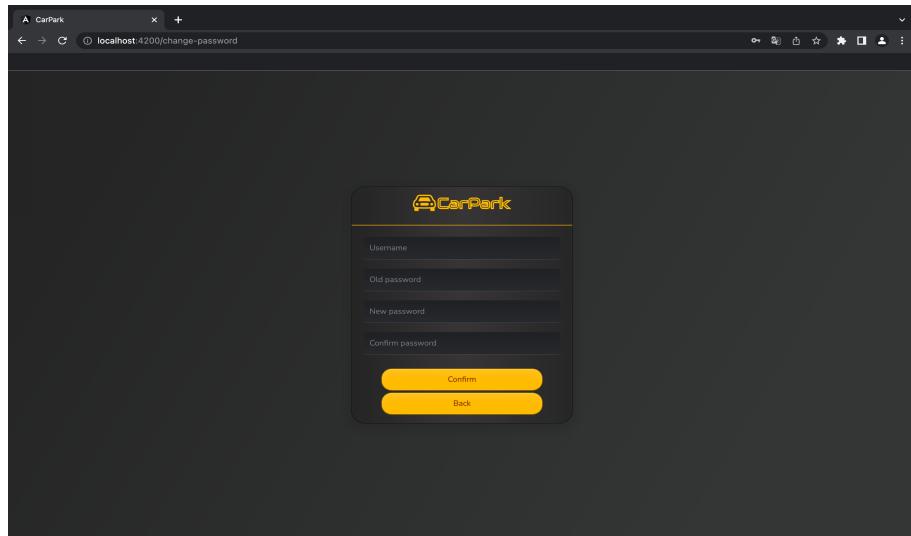


(b) Sikertelen regisztráció

2.3. ábra. Regisztrációs felület

2.4.3. Jelszóváltoztatási felület

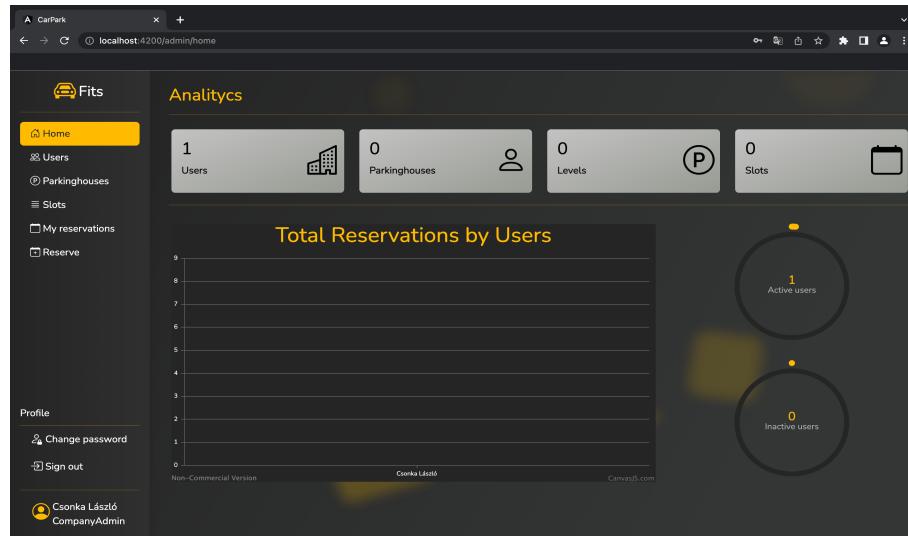
Amennyiben a felhasználó rendelkezik bármilyen jogosultságú fiókkal, akkor lehetősége van az azt védő jelszó megváltoztatására bejelentkezés után.



2.4. ábra. Jelszóváltoztatási felület

2.4.4. Home felület

A felhasználó ezt a felületet csak abban az esetben érheti el, ha a jogosultsága SystemAdmin vagy CompanyAdmin. A felület különböző diagramokat és információkat tartalmaz a parkolóházakról, foglalásokról és felhasználókról. A felhasználó láthatja az aktív és nem aktív felhasználói fiókok számát. Összesített adatot mutat az eddig regisztrált cégekről, felhasználókról és parkolóházakról.



(a) CompanyAdmin Home felület

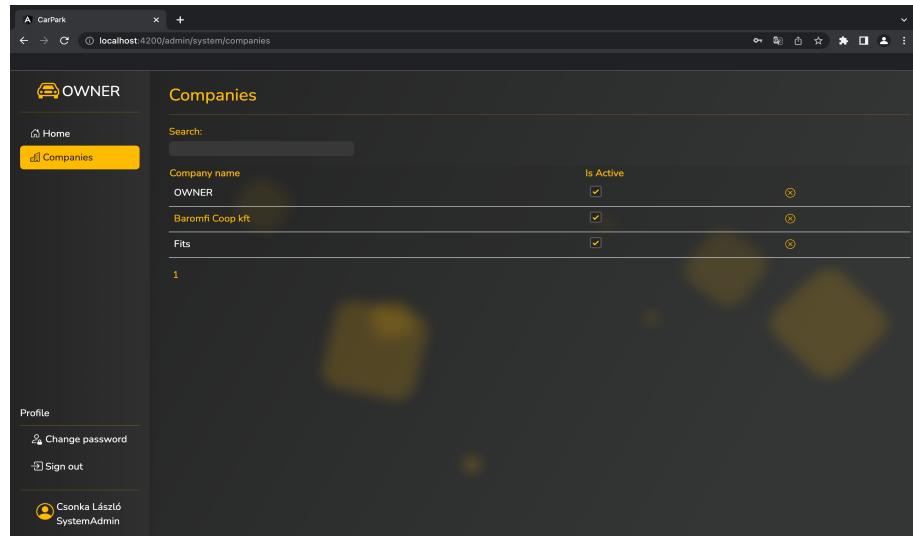


(b) SystemAdmin Home felület

2.5. ábra. Home felület

2.4.5. Companies felület

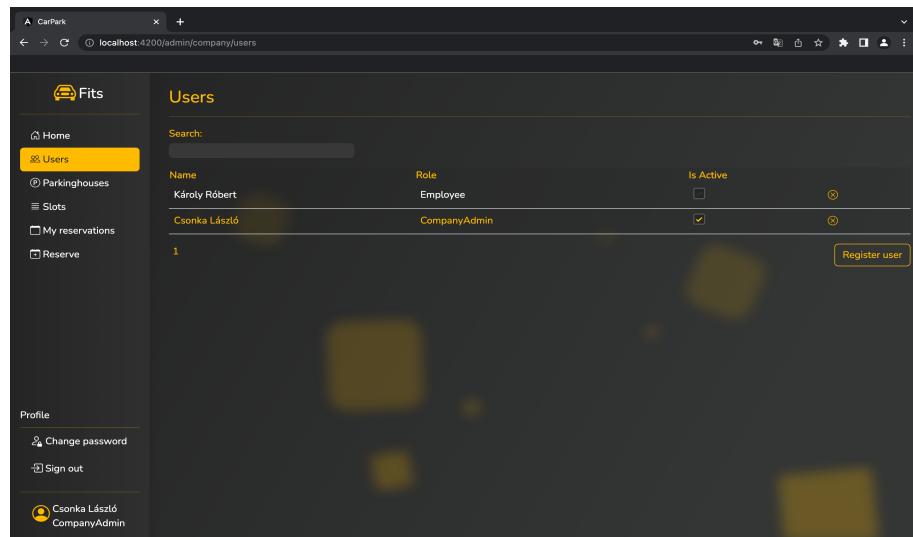
A felhasználó ezt a felületet csak akkor érheti el, ha SystemAdmin jogosultsággal rendelkezik. A felület a regisztrált cégek kezelését biztosítja táblázatos formában. A felhasználó itt aktiválhatja, deaktiválhatja, akár törölheti is azokat.



2.6. ábra. *Companies* felület

2.4.6. Users felület

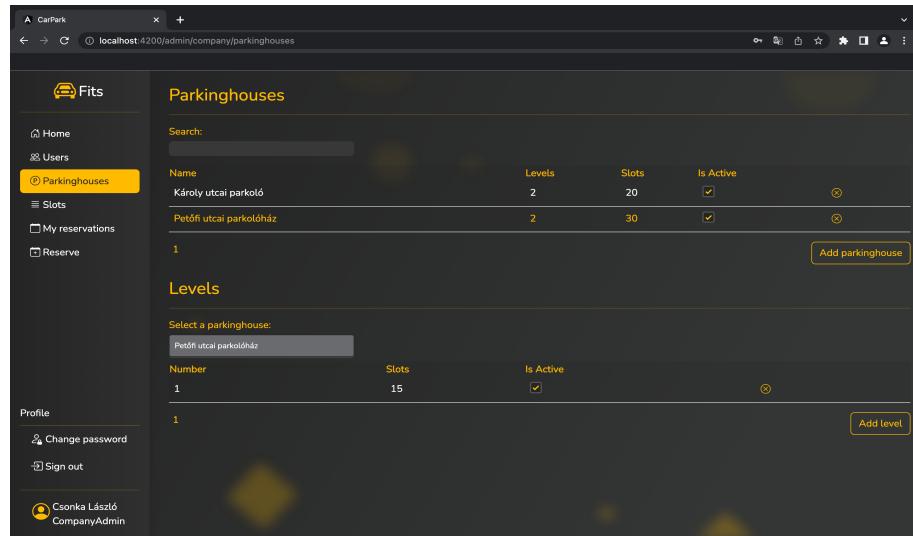
A CompanyAdmin jogosultsággal rendelkező felhasználók érik el ezt a felüleletet. A felület a felhasználók hozzáadására, törlésére, valamint aktiválására és deaktiválására biztosít lehetőséget egy táblázatban.



2.7. ábra. *Users* felület

2.4.7. Parkinghouses felület

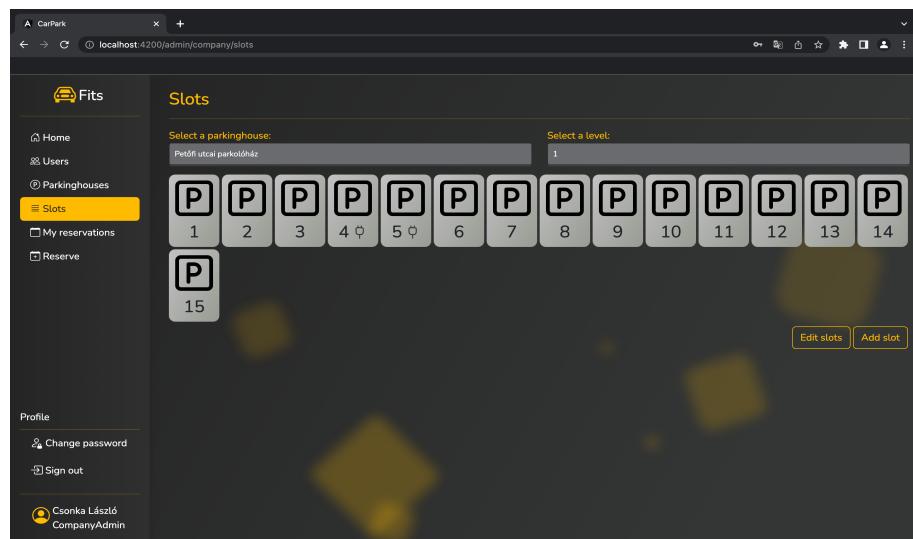
A CompanyAdmin jogosultsággal rendelkező felhasználók érik el ezt a felüleletet. A felület a parkolóházak és szintek hozzáadására, törlésére, valamint aktiválására és deaktiválására biztosít lehetőséget egy táblázatban.



2.8. ábra. *Parkinghouses* felület

2.4.8. Slots felület

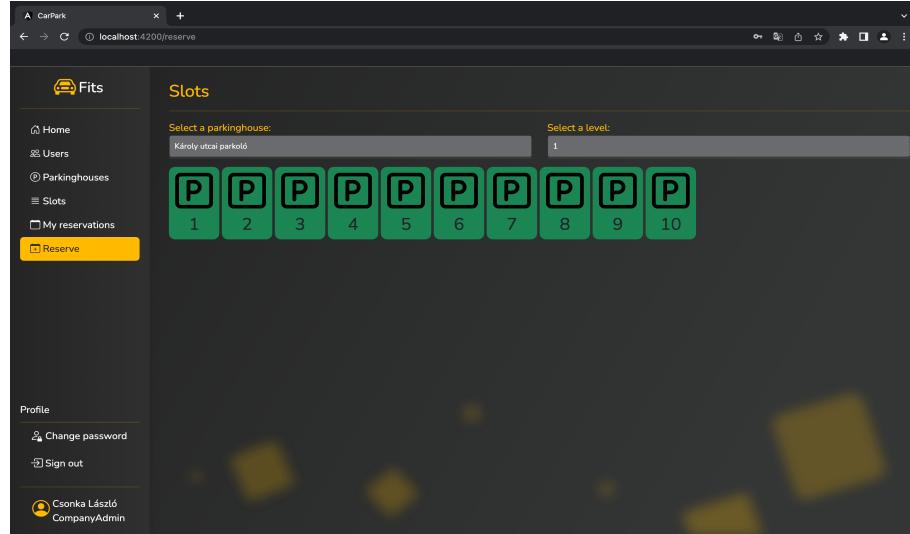
A CompanyAdmin jogosultsággal rendelkező felhasználók érik el ezt a felüleletet. Ezen a felületen a felhasználó a már korábban beállított parkolóhelyeket szabhatja személyre miután kiválasztotta, hogy melyik szintet szeretné személyre szabni. A felhasználónak lehetősége van egyszerre akár több férőhely törlésére vagy annak típusának módosítására is.



2.9. ábra. *Slots* felület

2.4.9. Reserve felület

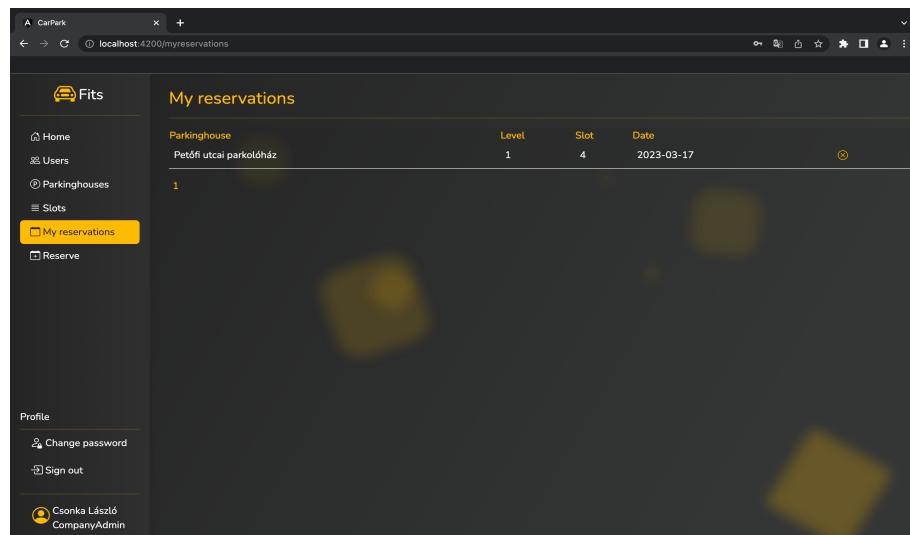
A SystemAdmin jogosultsággal rendelkező felhasználókon kívül bármelyik felhasználó eléri ezt a felüleletet, ahol végrehajthatja a következő foglalását.



2.10. ábra. *Reserve* felület

2.4.10. My reservations felület

A SystemAdmin jogosultsággal rendelkező felhasználókon kívül bármelyik felhasználó eléri ezt a felüleletet, ahol megtekintheti korábbi foglalásait, valamint törlheti, a még le nem záródott foglalásait.

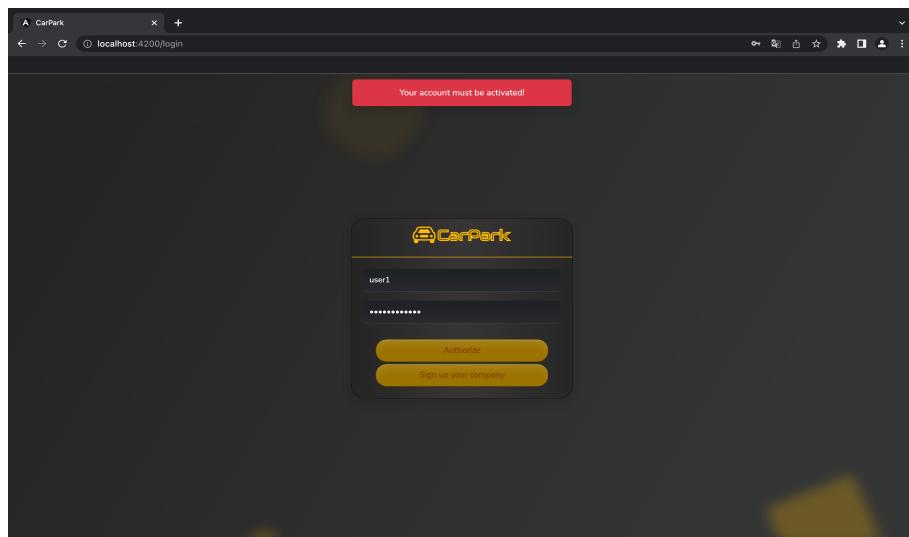


2.11. ábra. *My reservations* felület

2.5. Az alkalmazás használata

2.5.1. Bejelentkezés

A bejelentkezési felületnél látott sikeres és sikertelen bejelentkezés (2.2. ábra) nem csak attól függ, hogy a felhasználó a megfelelő felhasználónévét és jelszót gépelte-e be, hanem attól is, hogy az adott cég vagy az adott cégen belüli személy engedélyezve van-e a rendszeren belül. Az oldal ebben az esetben is megfelelő tájékoztatást nyújt a felhasználónak.



2.12. ábra. Nem engedélyezett cég vagy felhasználó esete

2.5.2. Regisztráció

Regisztrációkor egy CompanyAdmin jogosultságú felhasználó jön létre. A sikeres regisztráció érdekében minden mező kitöltése szükséges (2.3. ábra). A jelszónak tartalmaznia kell legalább egy nagybetűt és egy számot, valamint minimum 8 karakter hosszúságú kell, hogy legyen. Amennyiben valamelyik mező nem felel meg az elvárásoknak, az oldal tájékoztatja a felhasználót, hogy melyik mezőnél lépett fel hiba és hogy miért.

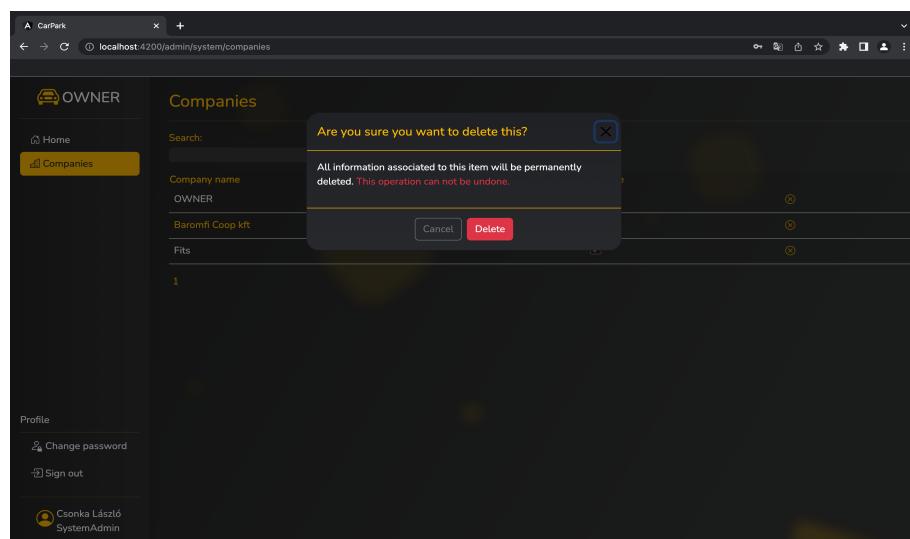
2.5.3. Cégek, felhasználók, parkoloházak, szintek aktiválása és deaktiválása

A Companies, Users, Parkinghouses felületeken látható táblázatokban, a felhasználó a jelölő négyzetek be- és kipipálásaval aktiválhatja és deaktiválhatja a cé-

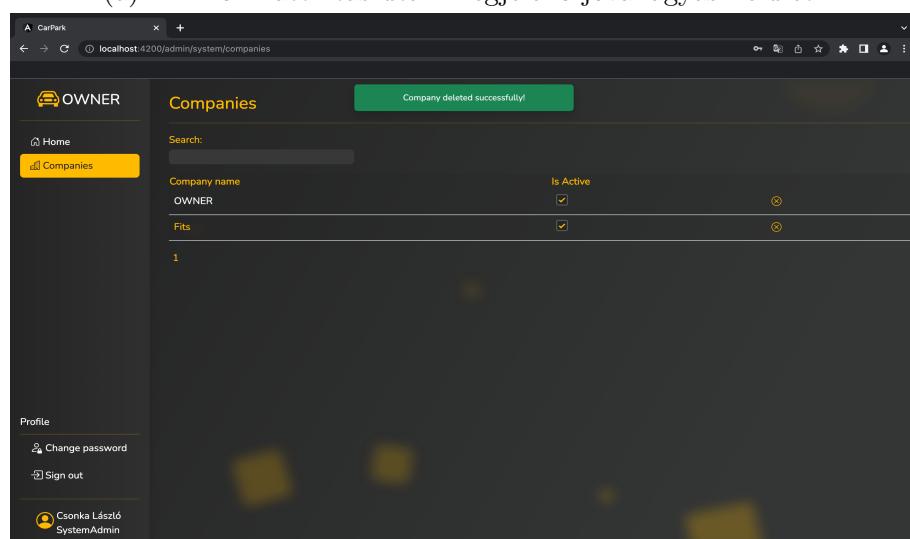
geket, felhasználókat, parkolóházakat és szinteket. A folyamat automatikusan végrehajtódik, amint a felhasználó interakcióba lép a jelölő négyzetekkel.

2.5.4. Cégek, felhasználók, parkolóházak, szintek eltávolítása

A Companies, Users, Parkinghouses, My reservations felületeken található táblázatok utolsó oszlopaiban egy kis 'X' ikon jelenik meg, amelyre ha a felhasználó rákattint, akkor egy megerősítésre váró ablak jelenik meg, ahol véglegesíthetjük az eltávolítást. A rendszer a sikeres eltávolításról tájékoztatja a felhasználót, valamint rögtön frissíti a felületen szereplő adatokat.



(a) 'X' ikon kattintás után megjelenő jóváhagyási felület

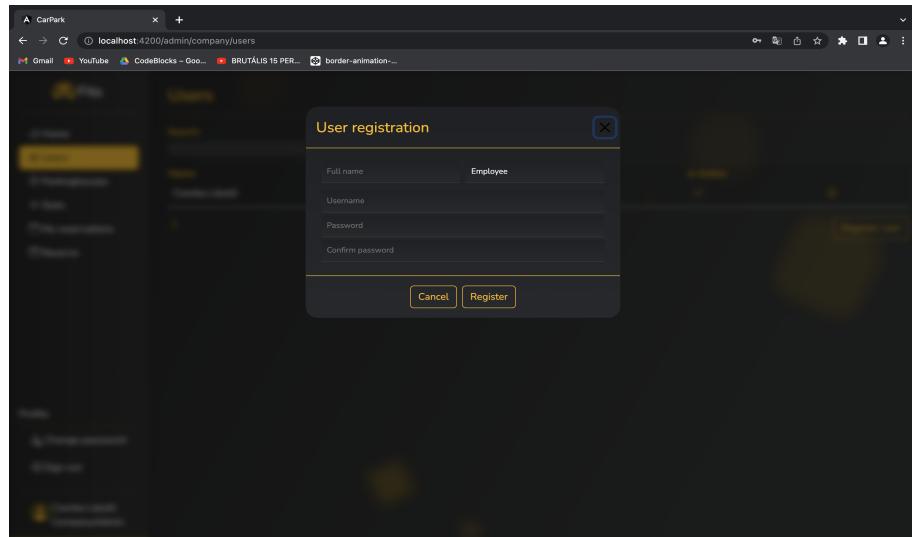


(b) Sikeres eltávolítás

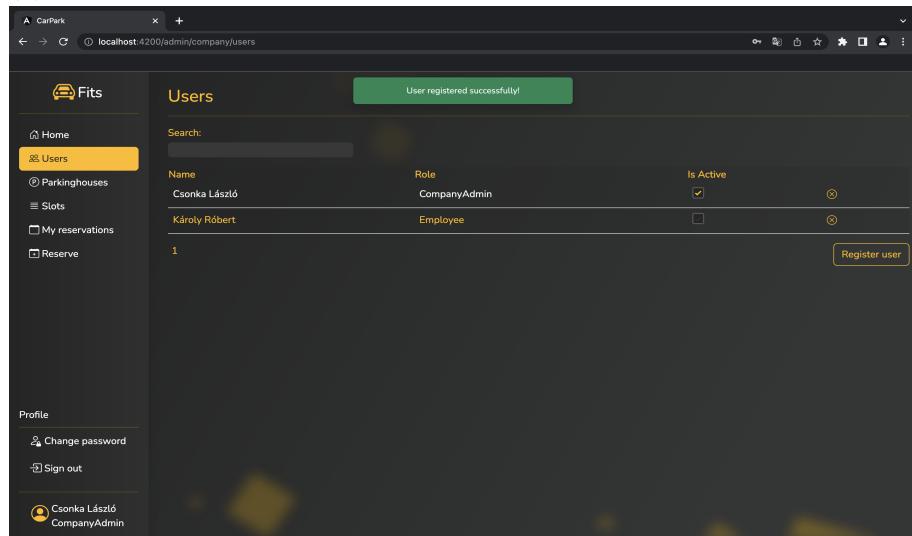
2.13. ábra. Eltávolítás

2.5.5. Felhasználók felvétele

A Users felületen található Register user gombra kattintva, a felhasználó egy regisztrációs ablakkal találkozik, amely ugyanúgy működik, mint a fentebb említett regisztráció esetében. Annyi különbséggel, hogy a cégnév helyett az adott felhasználó pozícióját kell beállítanunk egy legördülő menü segítségével. A felvétel sikeressége után, a táblázatban azonnal megjelenik a felhasználó által rögzített új felhasználó.



(a) A Register user gomb kattintás után megjelenő regisztrációs felület



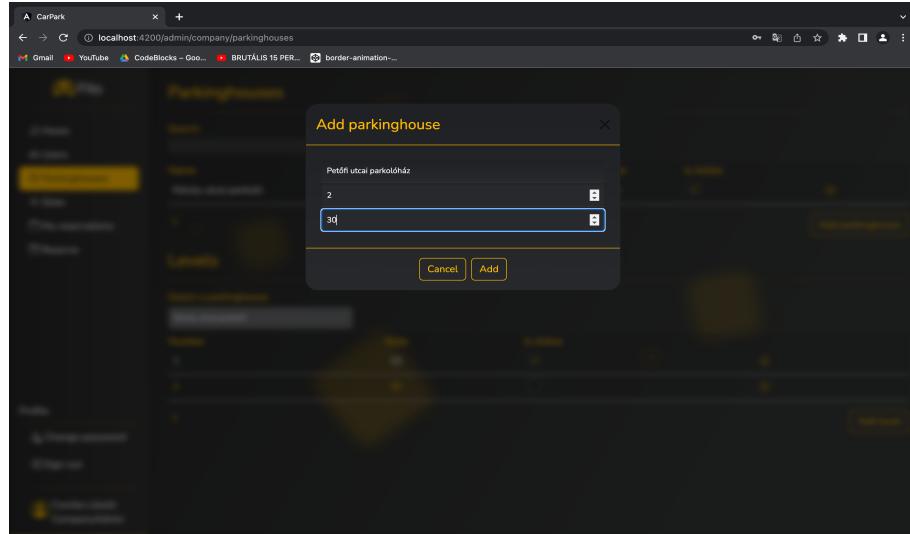
(b) Sikeres felvétel

2.14. ábra. Felhasználók felvétele

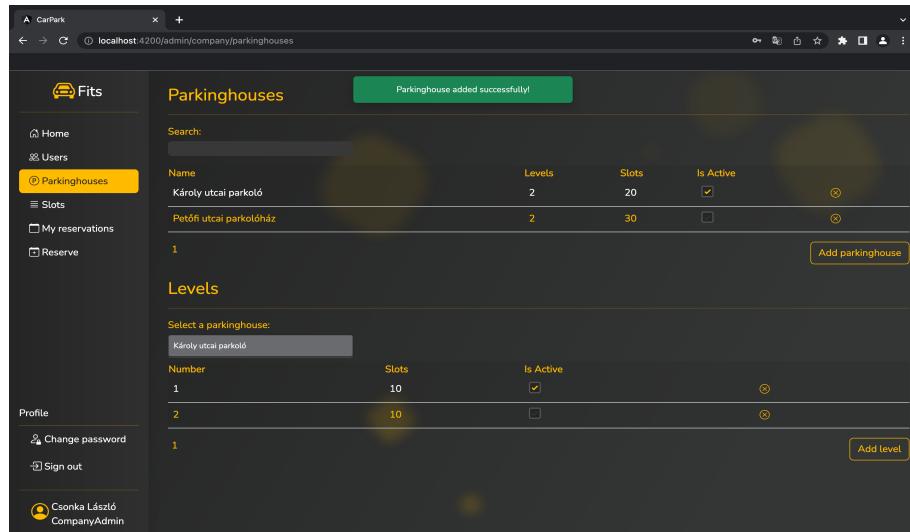
2.5.6. Parkoloházak felvétele

A Parkinghouses felületen a parkinghouses táblázat alatt szereplő Add parkinghouse gombra kattintva a felhasználó egy ablakkal találkozik,

ahol a parkolóház nevét, szintjeinek és férőhelyeinek számát kell megadni. Az ablakban minden mező kitöltése szükséges a sikeres felvétel érdekében, ezekről a felhasználót az oldal minden esetben tájékoztatja. Sikeres felvétel esetén az adatok automatikusan frissülnek a táblázatban.



(a) Az *Add parkinghouse* gomb kattintás után megjelenő ablak



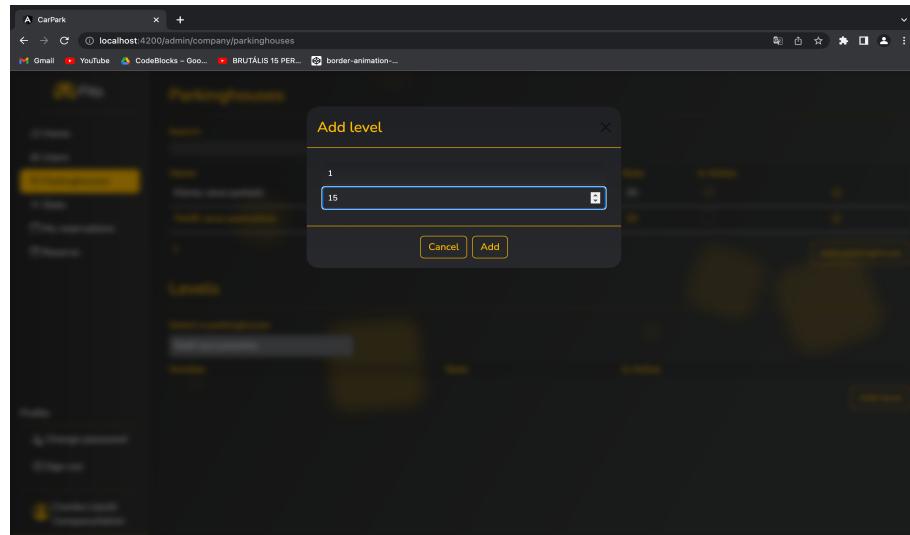
(b) Sikeres felvétel

2.15. ábra. Parkolóházak felvétele

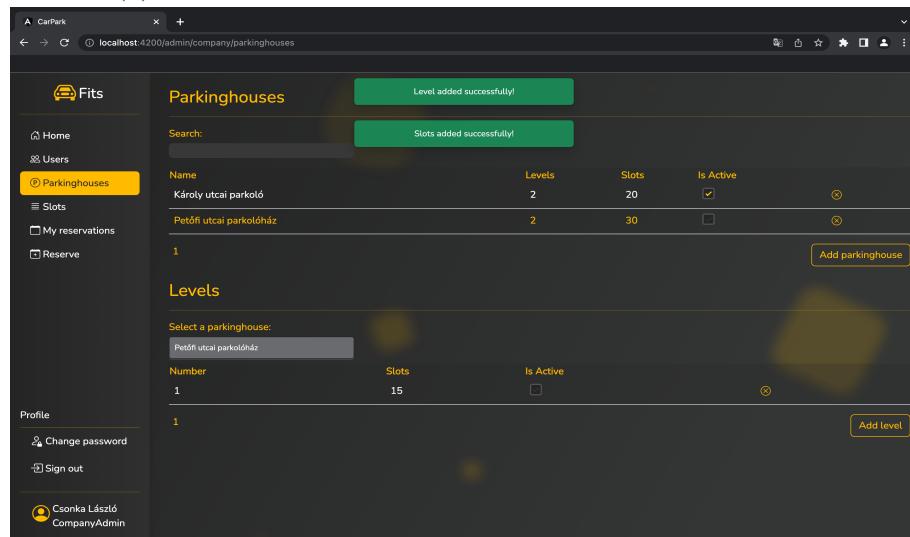
2.5.7. Szintek felvétele

A **Parkinghouses** felületen, a **Levels** táblázatban szereplő legördülő menüben választhatja ki a felhasználó az érintett parkolóházat. Kiválasztás után a táblázat alatt megjelenik egy **Add level** gomb, amelyre kattintva a felhasználó egy felvételi ablakkal találkozik. A **Level number** mező a szint számát jelöli, a **Slots** pedig,

hogy kezdetben hány ferőhelyet szeretnénk generálni, ugyanis az itt megadott Slots mező tartalma alapján a szinten automatikusan hozzáadásra kerülnek a férőhelyek. Az ablakban szereplő összes mező kitöltése kötelező a sikeres felvétel érdekében. Ezekről folyamatos tájékoztatást nyújt az oldal és sikeresség esetén nem csak egy visszajelző üzenettel találkozik a felhasználó, hanem a táblázatban szereplő adatok közvetlen frissülésének köszönhetően ott is megtalálható lesz az új szint.



(a) Az *Add level* gomb kattintás után megjelenő ablak



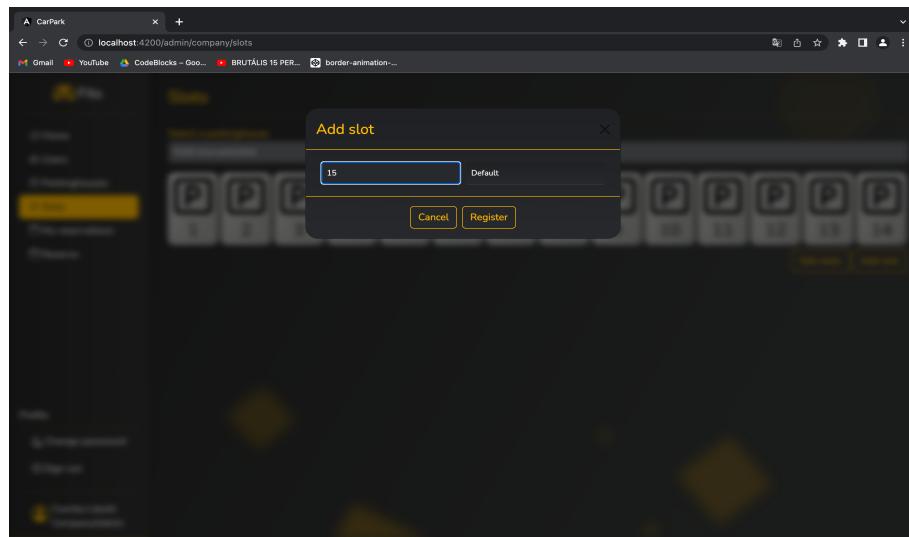
(b) Sikeres felvétel

2.16. ábra. Szintek felvétele

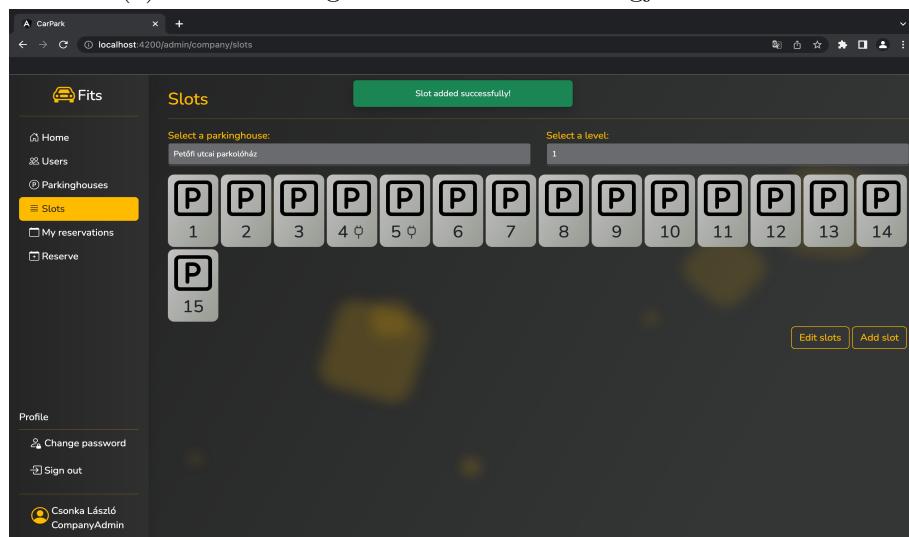
2.5.8. Parkolóhelyek felvétele

A Slots fülön a felhasználónak lehetősége van manuális helyfelvételre is, amelyet az Add slot gombra kattintva tehet meg, miután megfelelően kiválasztotta a parko-

lóhelyet és a szintet a legördülő menük segítségével. A kattintás után a felhasználó egy felvételi ablakkal fog találkozni, ahol a **Slot number**, azaz a hely számát és a típusát kell megadnia. A mezők kitöltése kötelező, a sikeres felvételt pedig az oldal egy üzenettel és automatikus hely frissítéssel is jelzi a felhasználónak.



(a) Az *Add slot* gomb kattintás után megjelenő ablak



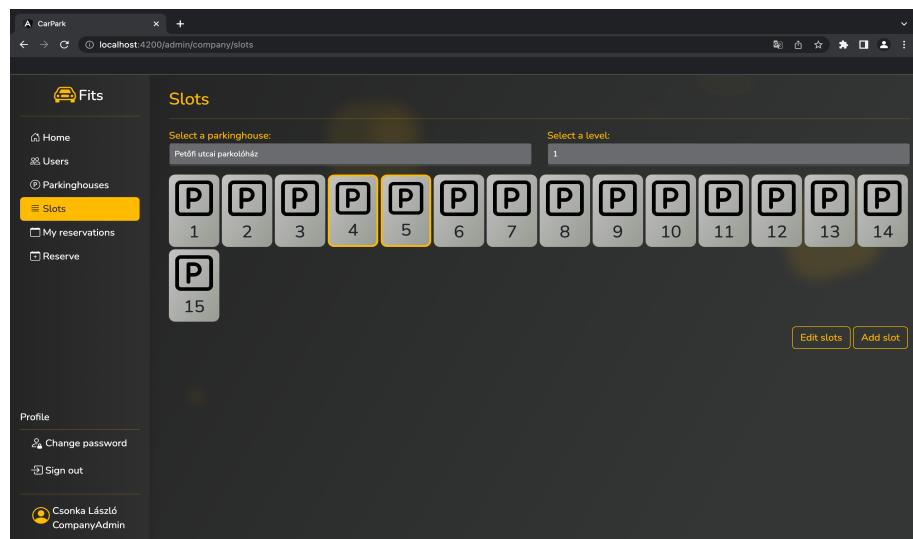
(b) Sikeres felvétel

2.17. ábra. Parkolóhelyek felvétele

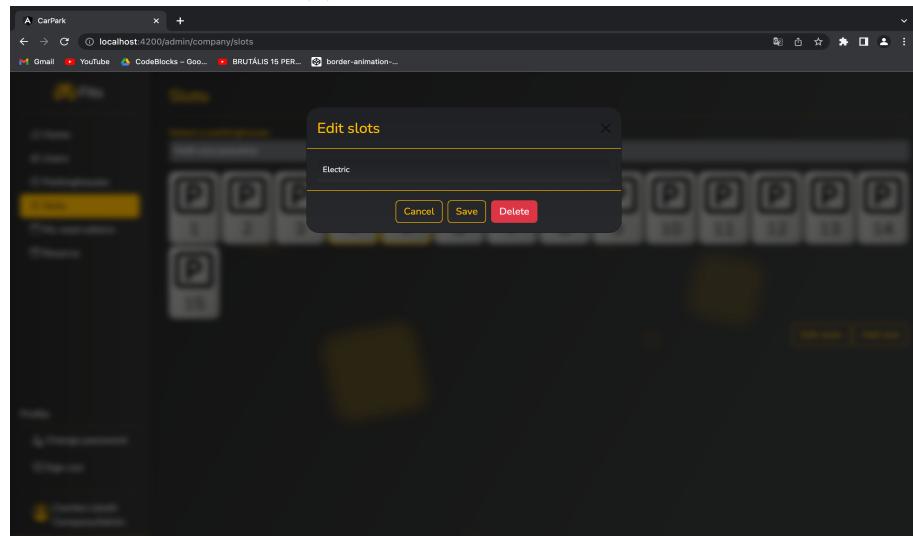
2.5.9. Parkolóhelyek személyre szabása

A **Slots** felületen a legördülő menükben a megfelelő parkolóházat és egy adott szintjét kiválasztva megjelennek az ott található férőhelyek. A férőhelyek megvannak számozva és a számuk mellett található kis ikonok jelzik a típusaikat, valamint a hagyományos parkolóhely esetében nem szerepel ikon. A korona az elsőbbségi, a

kerekesszék a rokkant, a csatlakozó pedig az elektromos parkolót jelöli. Kezdetben minden férőhely sima parkoló, azonban ezeket a felhasználó módosíthatja, amelyet a megfelelő helyek kijelölésével (kattintásával) tehet meg és a férőhelyek alatt szereplő **Edit slots** gombra kattintva kezdheti meg. Ezután a felhasználó egy szerkesztő ablakkal találkozik, ahol egy legördülő menüből kiválaszthatja az új típust és elmentheti azt a **Save** gombra kattintva. Amennyiben törölni kívánja a kiválasztott helyeket, akkor a szerkesztő ablakban csak annyi dolga van, hogy a **Delete** gombra kattint. A felhasználó minden művelet esetén visszajelzést kap az oldaltól.

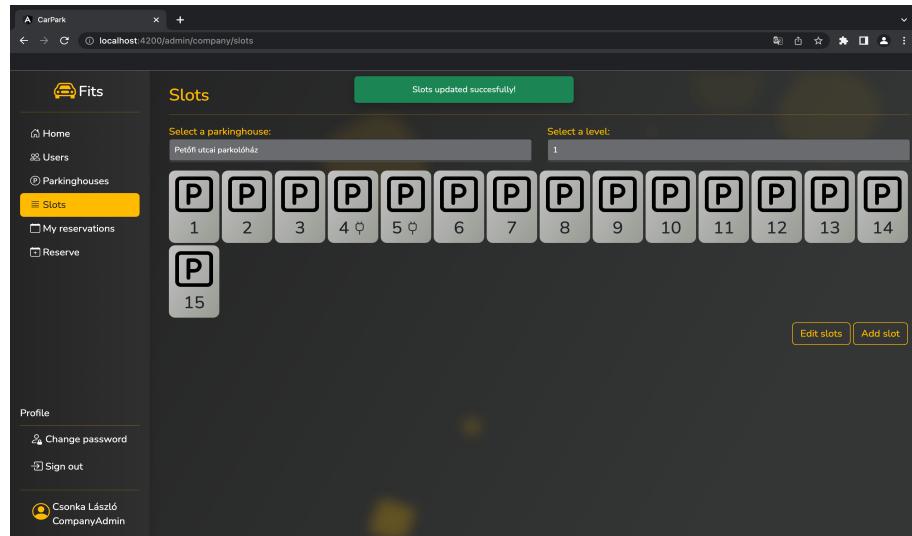


(a) A helyek kiválasztása

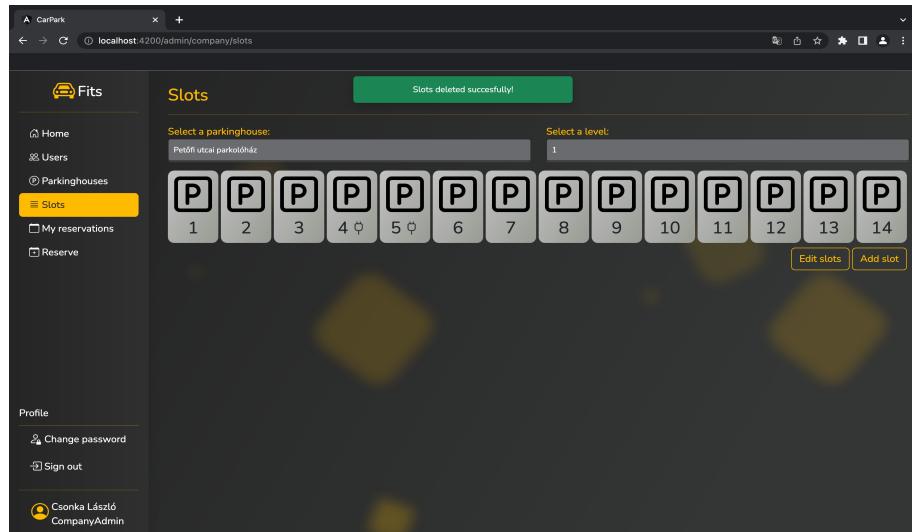


(b) Az *Edit slots* gomb kattintás után megjelenő ablak

2.18. ábra. Parkolóhelyek személyre szabása



(a) Sikeres frissítés



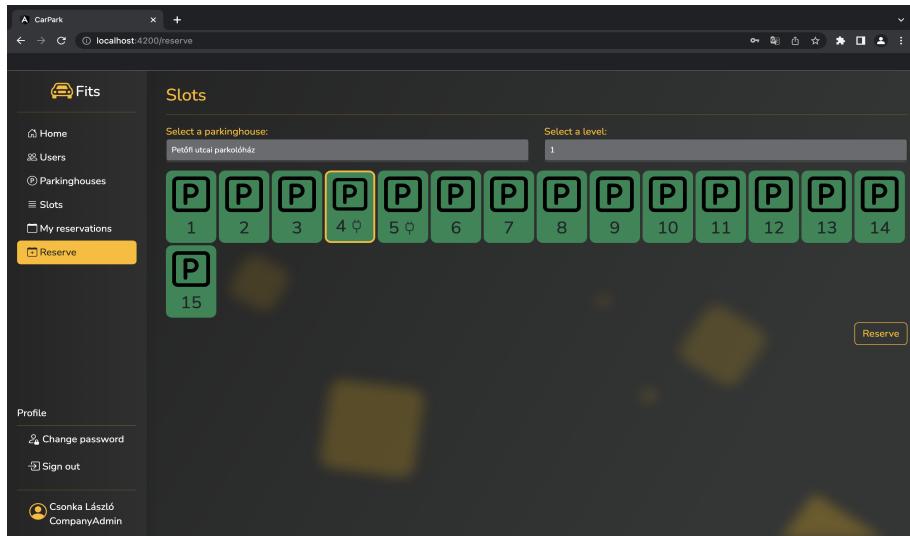
(b) Sikeres törlés

2.19. ábra. Parkolóhelyek személyre szabása

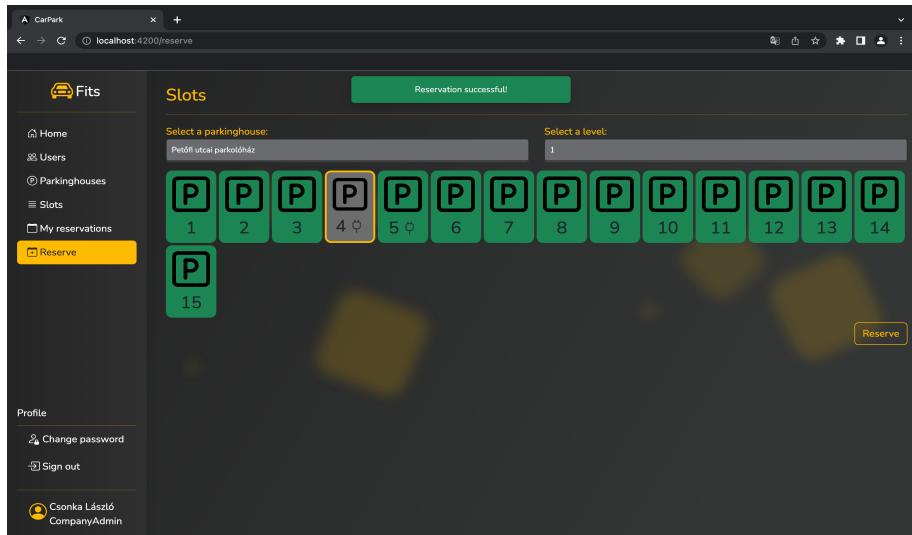
2.5.10. Foglalás menete

A Reserve felületen a legördülő menükből a felhasználó a megfelelő parkolóház és szint kiválasztása után láthatja a parkolóhelyeket. A parkolóhelyek színe jelzi azok státuszát: zöld esetén szabad, míg szürke esetén foglalt. A felhasználó az általa szimpatikus helyre kattintva kiválaszthatja azt és a helyek alatt megjelenő Reserve gombra kattintva véglegesítheti foglalását. Az elsőbbségi parkolók foglalására csak Boss jogosultsággal rendelkező felhasználóknak van lehetősége. A foglalás sikerességeről, esetleges sikertelenségének okáról az oldal tájékoztatja a felhasználót.

2. Felhasználói dokumentáció



(a) A hely kiválasztása



(b) Sikeres foglalás

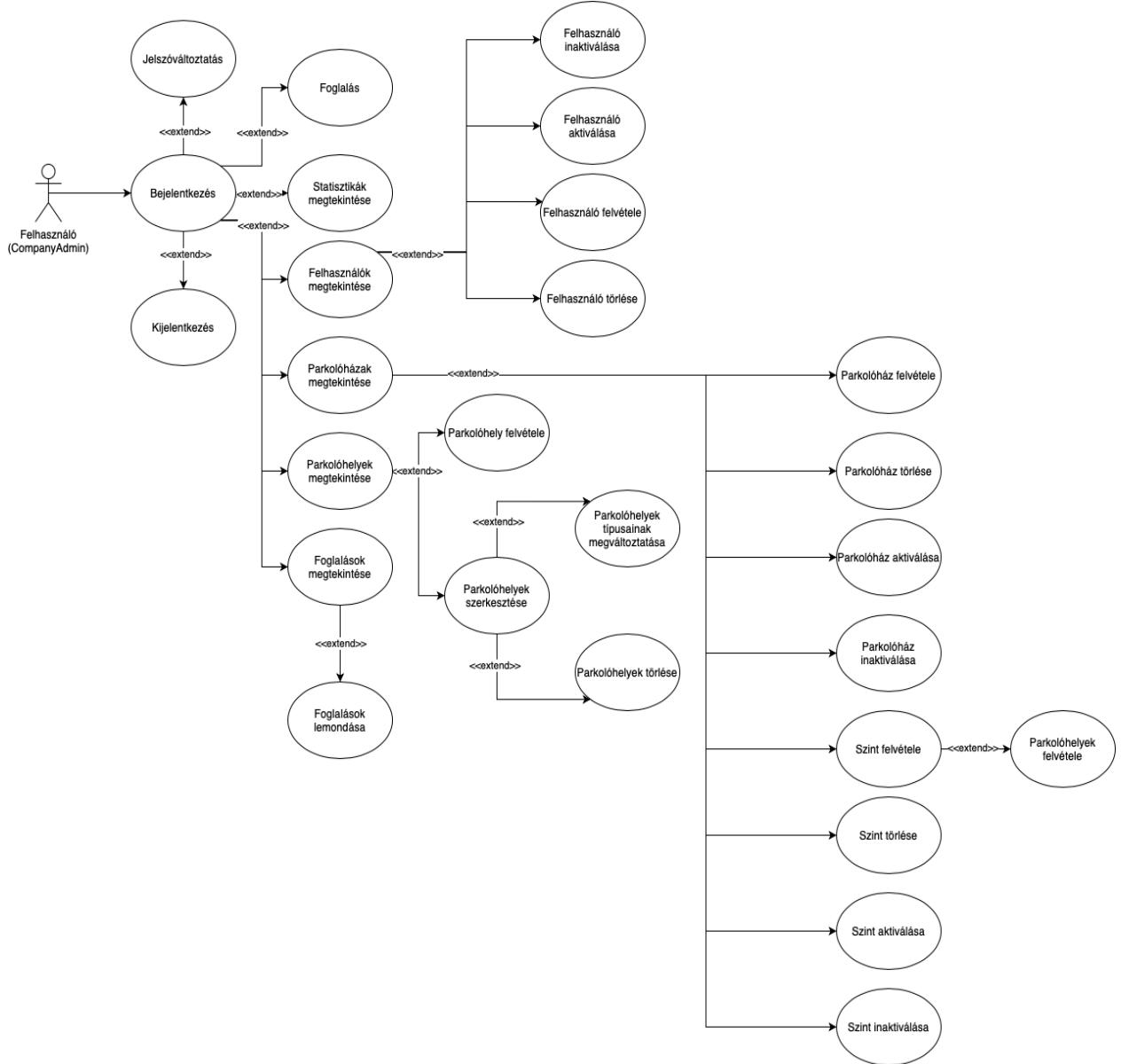
3. fejezet

Fejlesztői dokumentáció

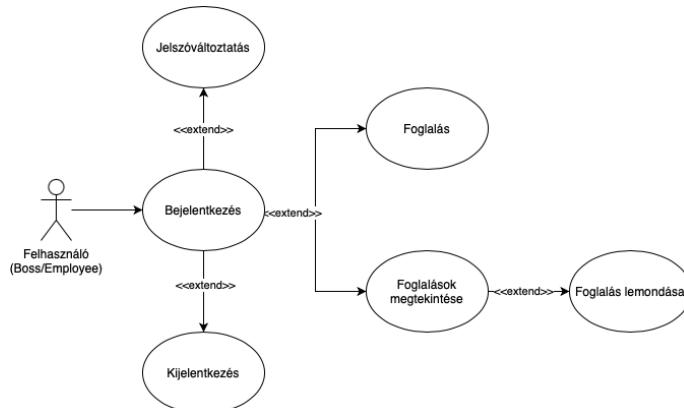
3.0.1. Tervezés és specifikáció

A cél egy olyan webalkalmazás létrehozása, amely egy könnyen kezelhető felületen lehetővé teszi a felhasználóknak, hogy online hajthassák végre a parkolás foglalását. Az alkalmazásnak támogatnia kell az összes leggyakrabban használt bőngészőt, és biztonságos módon kell tárolnia az adatokat, hogy ellenálljon a különféle támadásoknak. Az alkalmazás funkcionális követelményei az alábbiak:

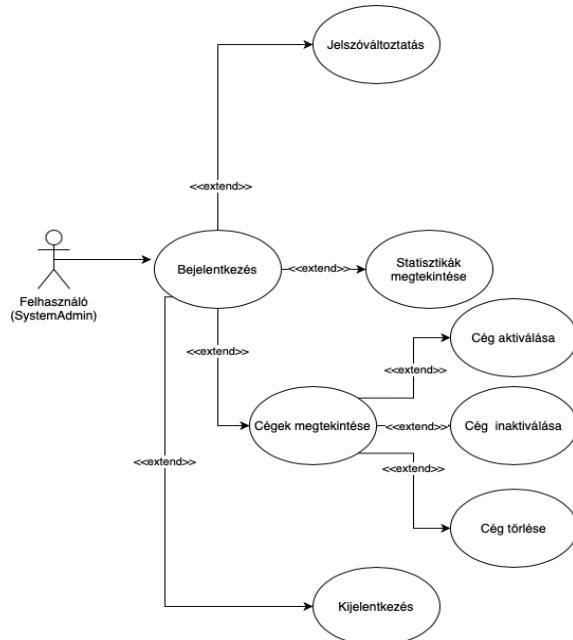
- Biztosítson egy könnyen használható felületet;
- Adjon lehetőséget az adatok áttekintésére, módosítására;
- Biztosítsa az adatok biztonságos tárolását;
- Lehetővé tegye a következő adatok kezelését:
 - Cégek
 - Felhasználók
 - Parkolóházak
 - Szintek
 - Parkolóhelyek
 - Foglalások



3.1. ábra. *CompanyAdmin* felhasználói esetdiagram



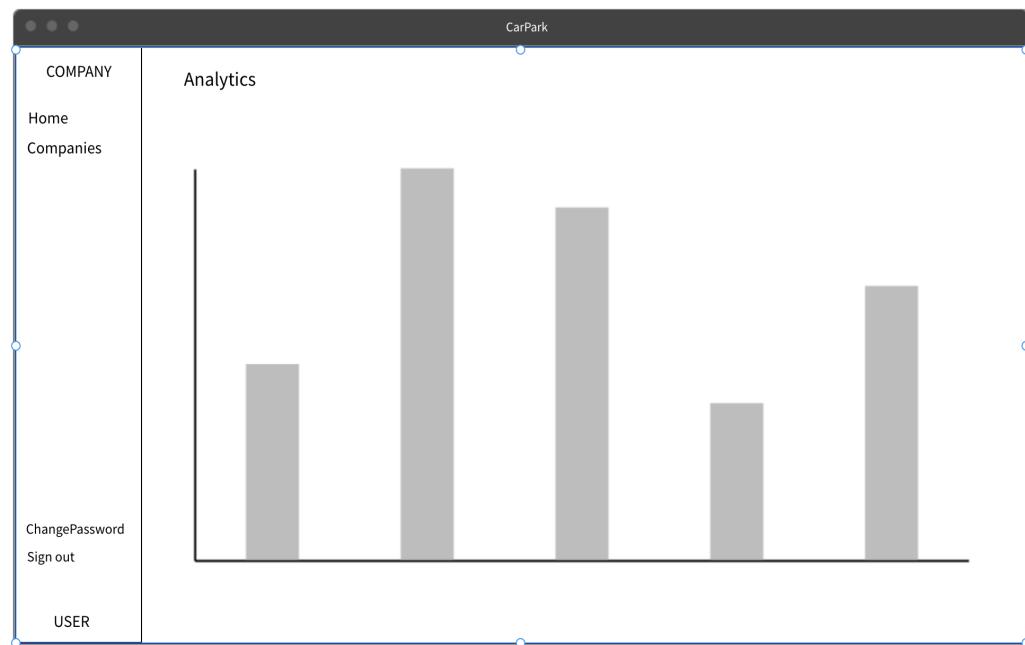
3.2. ábra. *Boss/Employee* felhasználói esetdiagram



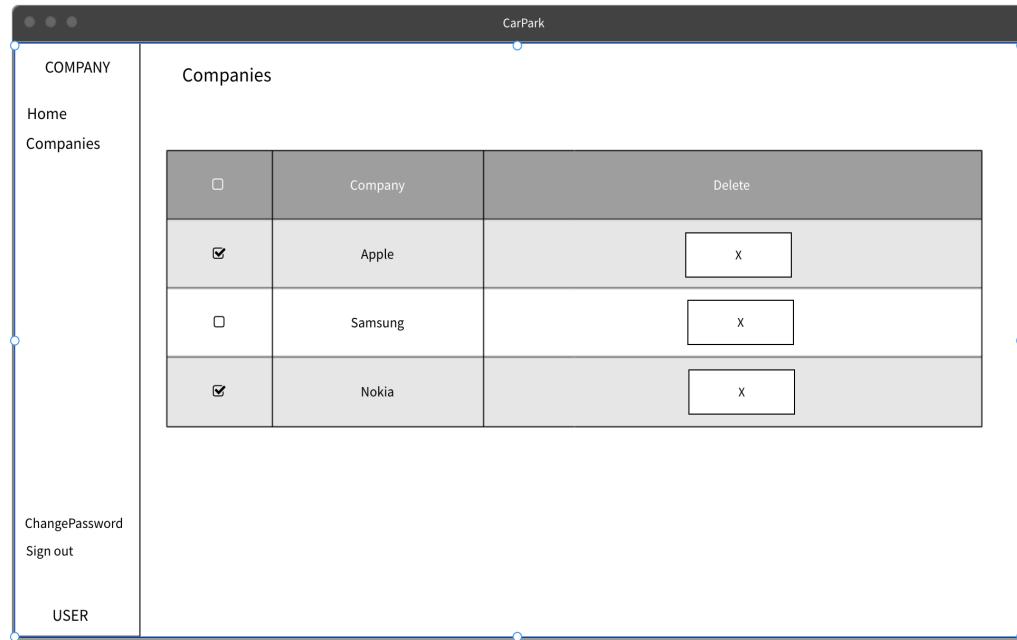
3.3. ábra. *SystemAdmin* felhasználói esetdiagram

3.1. Felhasználói felület

A webalkalmazás több nézetből áll, amelyek a Bootstrap[2] keretrendszerre épülnek. Az alkalmazás készítésének megkönnyítéséhez az alkalmazás főbb nézeteihez drótvázterv készült.

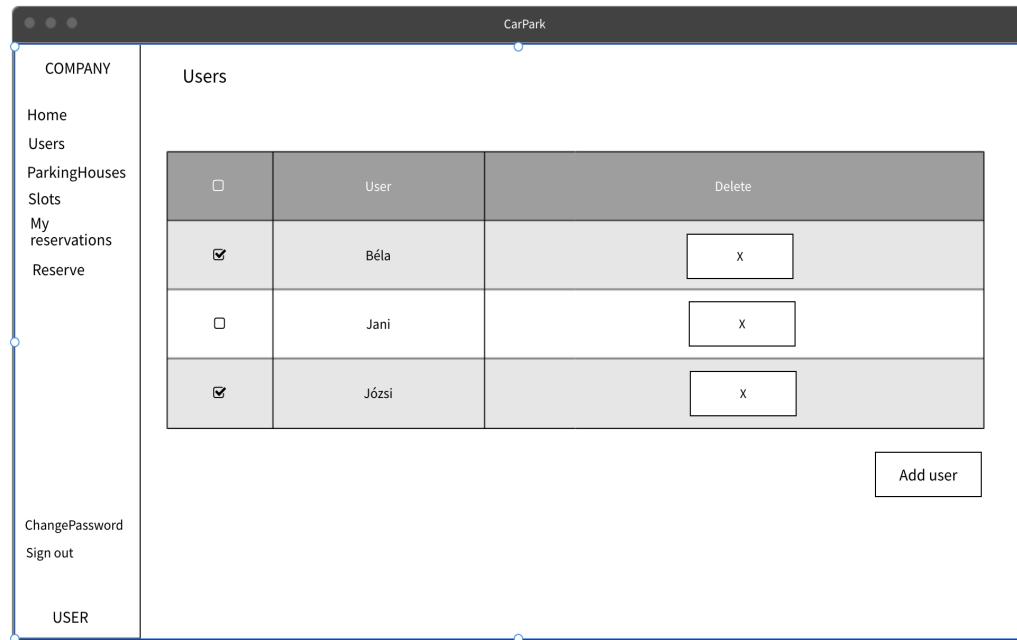


3.4. ábra. *Home* felületek drótvázterve



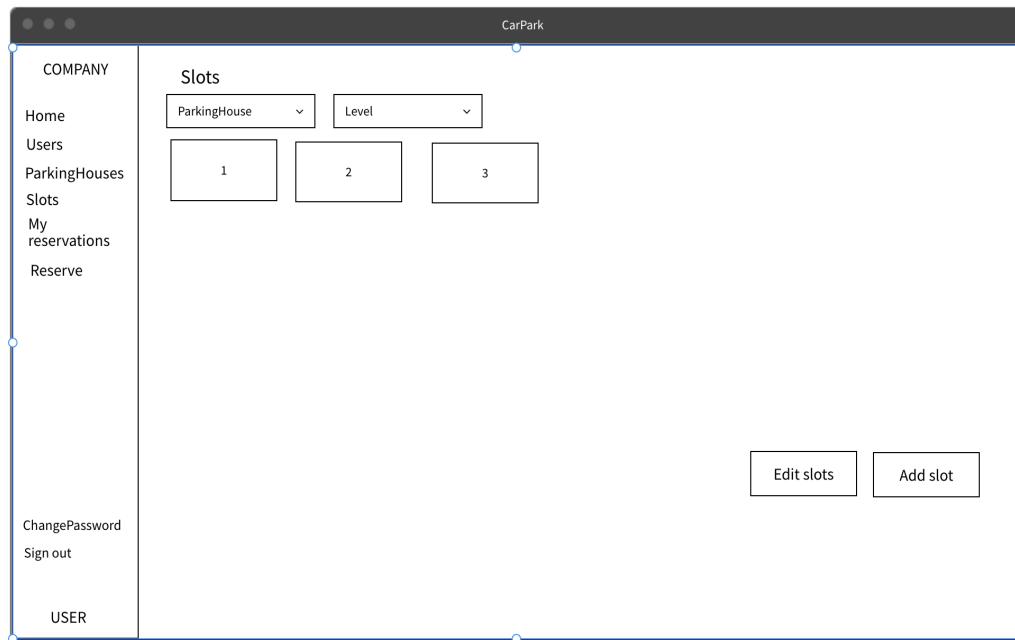
3.5. ábra. *Companies* felület drótvázterve

A *Users* felület hasonló komponenseket és felépítést használ, mint a *Parkinghouses* felület, a különbség annyi, hogy két táblázat helyezkedik el azon a fülön, az egyik a parkoloházaké, a másik pedig a szinteké.



3.6. ábra. *Users* felület drótvázterve

A *Slots* felület hasonló komponenseket és felépítést használ, mint a *Reserve* felület, annyi különbséggel, hogy a *Reserve* fülön csak egyetlen foglalás gomb található.



3.7. ábra. *Slots* felület drótvázterve

A *My reservations* fül egyszerűen csak egy táblázat komponenst tartalmaz.

3.2. Fejlesztői környezet konfigurálása

3.2.1. Felhasználói felület fejlesztői környezet konfigurálása

A fejlesztéshez bármilyen rendszer használható, melyre elérhető a Node.js[3]. Az ajánlott fejlesztői környezet a Visual Studio Code[4]. Az instrukciók Windows 10 és macOS Ventura operációs rendszeren lettek tesztelve.

Az alkalmazás futtatásához szükség van a Node.js-re, git[5] verziókezelőre, valamint az npm csomagkezelő alkalmazásra. A Node.js automatikusan telepíti az npm csomagot a hivatalos weboldalról elért telepítővel. Git esetében opcionálisan használhatunk grafikus Git klienst is. A továbbiakban feltételezzük, hogy a program forráskódja megtalálható az eszközünkön.

Előkészítés és futtatás

Az alkalmazás futtatásához szükséges könyvtárakat az npm csomagkezelő tudja telepíteni, ezt az `npm install` parancssal tehetjük meg. A szükséges könyvtárak listáját a `package.json` fájl tartalmazza, ezek alapján automatikusan letöltődik minden, amire szükség van.

A könyvtárak telepítésének befejezte után adjuk ki az `ng serve` parancsot, amely legenerálja nekünk az alkalmazás felhasználói felületét, a sikeres generálás után az oldal `http://localhost:4200` címen érhető el.

3.2.2. API végpontok fejlesztői környezet konfigurálása

A fejlesztéshez bármilyen rendszer használható, melyre elérhető a .NET 6[6] keretrendszer. A fejlesztői környezet a Visual Studio Community 2022[7]. Az instrukciók Windows 10 operációs rendszeren lettek tesztelve. A szoftver fejlesztéséhez szükség van a .NET keretrendszer 6-os verziójának fejlesztői változatára, valamint egy működő MSSQL adatbázis kapcsolatra, valamint git-re[5], amely akár grafikus kliens is lehet. A továbbiakban feltételezzük, hogy a program forráskódja megtalálható az eszközünkön.

Előkészítés és futtatás

A Visual Studio Community-ben[7] egyszerűen indítsuk el a programot, a program buildelése és futtatása után automatikusan megjelenik egy Swagger felület, ahol a végpontokat tudjuk tesztelni a felhasználói felület futtatása nélkül.

3.3. Alkalmazott technológiák

3.3.1. Angular

Az Angular[8] egy TypeScriptre épülő, nyílt forráskódú fejlesztési platform. Elsősorban frontend oldali fejlesztésre, összetett webalkalmazások megvalósítására találták ki. Az Angular alapja a komponens alapú architektúra, amely lehetővé teszi a felhasználói felület logikai és vizuális elkülönítését. Az Angular keretrendszer több platformon is használható.

3.3.2. Visual Studio Code

A Visual Studio Code[4] a Microsoft ingyenes, nyílt forráskódú multiplatform fejlesztői környezete, amely rengeteg hasznos integrációval és bővítménnyel rendelkezik.

3.3.3. Visual Studio Community 2022

A Visual Studio Community[7] a Microsoft ingyenes multiplatform fejlesztői környezete. .NET 6-hoz tökéletes, ugyanis nagymértékben hozzájárul a nagyobb projektek fejlesztéséhez.

3.3.4. Git, GitHub és GitHub Desktop

Az alkalmazás forráskódjának verziókezelése a Git szoftver segítségével történik. A Git[5] egy ingyenes, nyílt forráskódú verziókezelő rendszer, ami lehetővé teszi a közös, távoli munkát, és a fejlesztések követését. A projekt GitHub szerveren található. A lokális Git műveletek (commit, push, pull) az ingyenes GitHub Desktop-ban[9] és konzolos felületen történnek.

3.3.5. MSSQL

A webalkalmazás adatbázisban tárolja az adatokat. A fejlesztés során MSSQL[10] szervert használt az alkalmazás.

3.3.6. Draw.io

A Draw.io[11] tervezőszoftverben az alkalmazás különböző diagramjai valósultak meg.

3.4. Felhasznált keretrendszerök

3.4.1. Entity Framework

Az Entity Framework[12] egy programozási keretrendszer, amely objektumokat képes leképezni az adatbázisban található táblákra, és automatizált módon segíti a fejlesztőket az adatok tárolásában és elérésében. Kiváló megoldás Code First adatbázisok esetén is.

3.4.2. Identity Framework

Az Identity Framework[13] lehetővé teszi a felhasználók regisztrációját, bejelentkezését és jogosultságainak kezelését. A keretrendszer tartalmaz előre elkészített

modulokat a felhasználók kezeléséhez, például a jelszókezelőt, az e-mail megerősítést és a kétlépcsős hitelesítést.

3.4.3. Bootstrap

A Bootstrap[2] a legnépszerűbb CSS-keretrendszer a reszponzív és mobilbarát weboldalak fejlesztéséhez.

3.5. Alkalmazás megvalósítása

Az MVC[14] rövidítés a Model-View-Controller szavakból származik, és egy olyan architektúrális mintát jelent, amelyet szoftvertervezésben alkalmaznak. Az MVC segít a kód szétválasztásában, hogy az könnyebben karbantartható és bővíthető legyen.

A Model a szoftver adatmodelljét képviseli, és magába foglalja az adatok kezeléséért és azok lekérdezéséért felelős kód részleteket. A Model szerepe, hogy biztosítsa az adatok helyes működését és biztonságát.

A View az adatok megjelenítéséért felelős része a szoftvernek. A View tartalmazza az összes felhasználói felületi elemet, amelyekkel a felhasználó interakcióba léphet a szoftverrel.

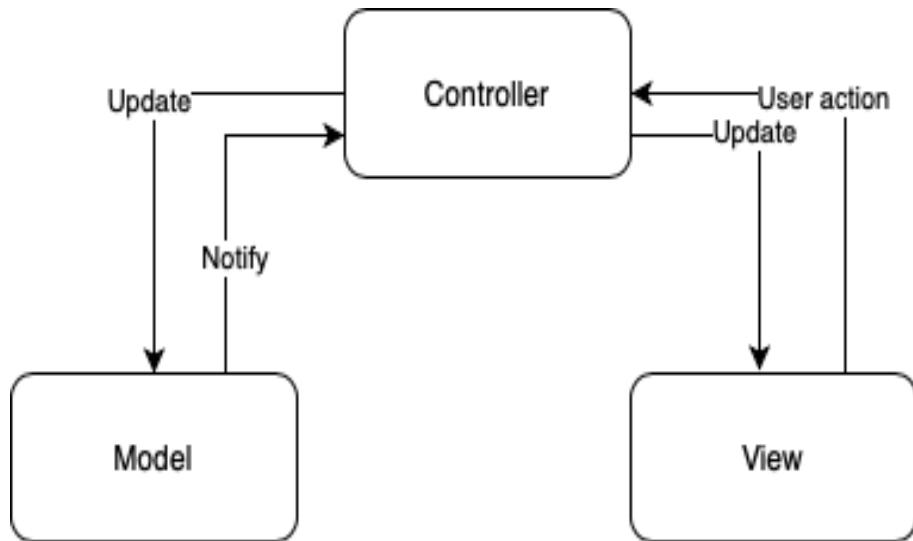
A Controller a Model és a View közötti kapcsolatot biztosítja. A Controller fogadja a felhasználói interakciókat, és eldönti, hogy melyik Model műveleteket kell végrehajtania. Azután frissíti a View-t, hogy megjelenítse az új adatokat, amelyeket a Model frissített.

Az MVC minta segítségével a szoftver komponensei egymástól elküllönülnek, így azokat külön-külön fejleszthetjük, karbantarthatjuk és tesztelhetjük.

Angular MVC architektúrát támogat, minden komponenséhez tartozik egy controller *typescript* fájl, ami háttérlogikákat végez, létrehozza a kapcsolatot a szerverrel, ami .NET alapú implementáció.

A frontend és backend oldalon azonos alapú modellek találhatóak, ez azért szükséges, hogy a két oldal megfelelően tudjon egymással kommunikálni, és ezt a kommunikációt a kontrollerek biztosítják. Külön kiemelném négy fajta controllerem: `SystemAdminController`, `CompanyAdminController`, `DefaultUserController`, melyek tükrözik az alkalmazásban megvalósuló jogosultságokat, valamint szerepel a programban egy `UserProfileController`, ami az autentikációért felel.

View felhasználva a kontrollereket kirendeli az ott eltárolt adatstruktúrákat.



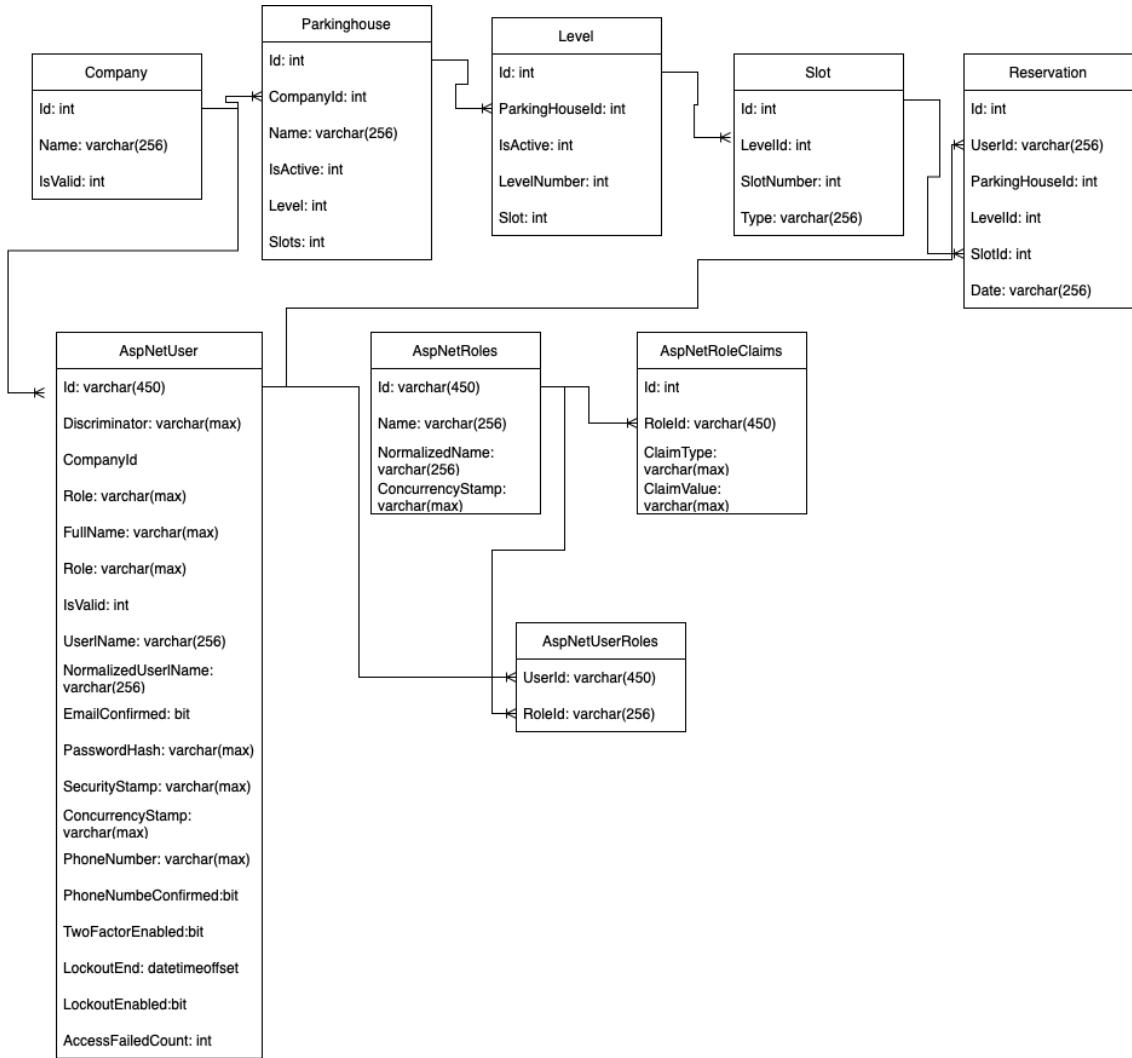
3.8. ábra. Csomag diagram

A jogosultságok kezelése az Identity[13] rendszerben történik, amely lehetővé teszi a felhasználói fiókok és az azokhoz kapcsolódó jogosultságok kezelését. Az alkalmazásban négy felhasználói csoport található (SystemAdmin, CompanyAdmin, Boss, Employee).

A Dependency Injection[15] lehetővé teszi hogy, különböző alrendszerök szolgáltatóként legyenek kezelve, és az ezek közötti függőségek automatikusan injektálódnak a kód futása során, a komponensek közötti kommunikáció biztosítása érdekében.

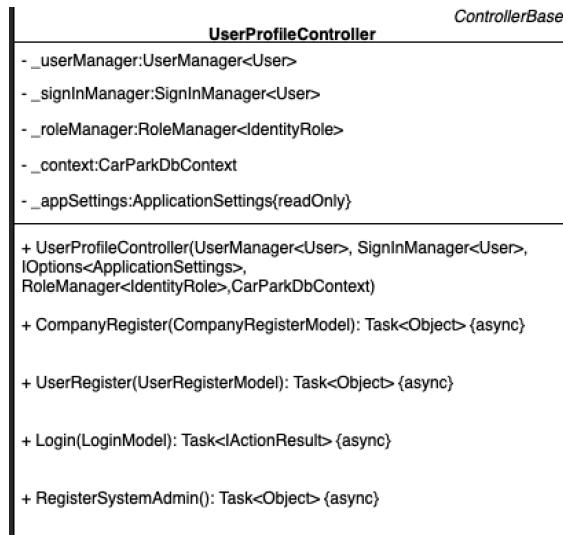
3.5.1. Adatbázis séma

Az alkalmazásban az Entity Framework Core 6 könyvtár felel az adatbázissal történő kommunikációért. Ez lehetővé teszi, hogy az adatmodell teljes mértékben a C# kódban legyen definiálva, és automatikusan létrehozza, illetve frissíti az adatbázis sémáját (Code First adatbázis). Az EF Core 6 egyszerűsíti az adatok olvasását és módosítását is, mivel lehetővé teszi a C# lista objektumokon keresztüli hozzáférést az adatbázisban tárolt táblákhoz. A lekérdezések LINQ (Language Integrated Query) szintaxis segítségével íródnak, amelyeket a könyvtár automatikusan átalakít SQL utasításokká, amelyeket az adatbázisban lehet futtatni. Az adatbázis struktúrája a következő ábrán látható.

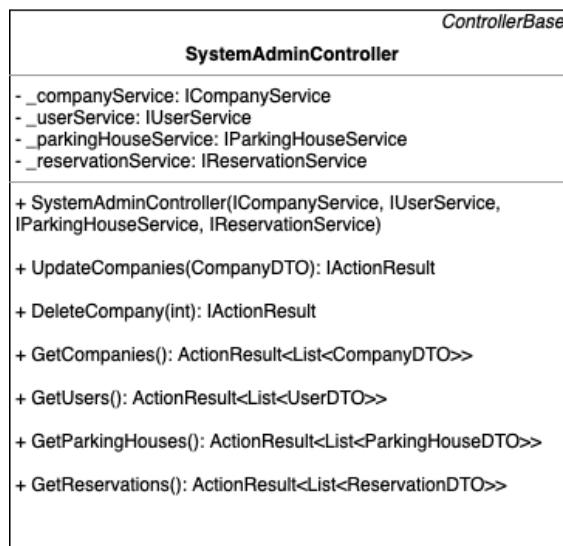


3.9. ábra. Adatbázis séma

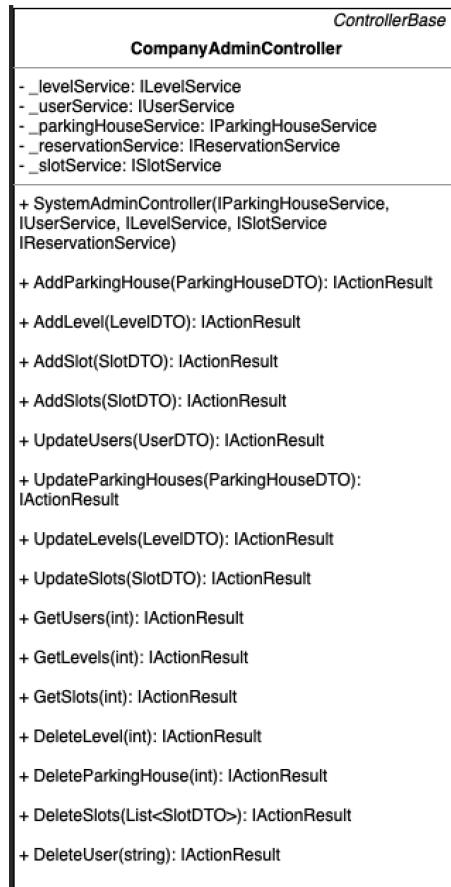
3.5.2. API végpontok osztálydiagramja



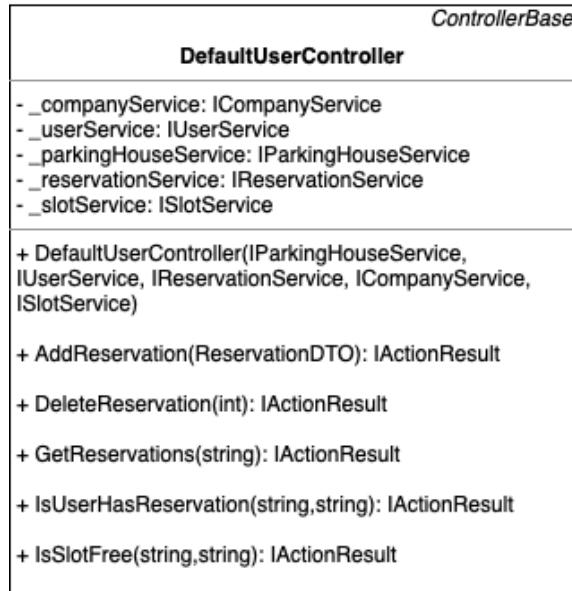
3.10. ábra. *UserProfile* osztálydiagramja



3.11. ábra. *SystemAdmin* osztálydiagramja



3.12. ábra. *CompanyAdmin* osztálydiagramja



3.13. ábra. *DefaultUser* osztálydiagramja

3.6. Tesztelési terv

A tesztelési folyamat két nagy részre bontható:

- Felhasználói felület tesztelése
- API végpontok tesztelése

3.6.1. Felhasználói felület tesztelése

A program kisebb komponensekből épül fel, ezért azok egységesztelésére a Cypress[16] tesztelő keretrendszer használtam. A tesztelés során minden főbb funkciót érintek. A tesztelési terv a következő táblázatban látható.

3.1. táblázat. Felhasználói felület tesztelése

Teszt	Teszteset	Elvárt eredmény
login spec	Megvizsgálja hogy a megfelelő form-ok és mezők megjelentek-e, majd beírja a bejelentkezési adatokat, és bejelentkezik.	A bejelentkezés sikeres vagy sikertelen.
register spec	Megvizsgálja hogy a megfelelő form-ok és mezők megjelentek-e, majd begépeli a szükséges adatokat és végrehajta a regisztrációt.	A sikeres regisztráció után a bejelentkezási felületre navigál.
change-password-cancel spec	Megvizsgálja, hogy a megfelelő form-ok és mezők megjelentek-e, majd visszalép.	A jelszóváltoztatást elveti és visszatér a főmenübe.
change-password spec	Megvizsgálja, hogy a megfelelő form-ok és mezők megjelentek-e, majd begépeli a szükséges adatokat és végrehajta a jelszóváltoztatást.	A jelszóváltoztatás sikeresen végrehajtódik és visszatér a főmenübe.
navbar-navigation spec	Megvizsgálja, hogy a megfelelő jogosultságoknál a megfelelő menüpontok jelennek-e meg.	Sikeresen megjelennek az elvárt menüpontok.

simple-user spec	Megvizsgálja, hogy a menü megjelent-e és végigjárja a menüpontokat.	A menüpontokra kattintva a kért fülek sikeresen megjelennek, majd a felhasználó kilép.
company-admin-user spec	Megvizsgálja, hogy a menü megjelent-e és végigjárja a menüpontokat.	A menüpontokra kattintva a kért fülek sikeresen megjelennek, majd a felhasználó kilép.
system-admin-user spec	Megvizsgálja, hogy a menü megjelent-e és végigjárja a menüpontokat.	A menüpontokra kattintva a kért fülek sikeresen megjelennek, majd a felhasználó kilép.
add-user-modal spec	A Users fülre navigál. Megvizsgálja, hogy megjelentek-e a szükséges elemek, majd megnyitja a szint hozzáadói modalt. Majd először az Register gombra, majd a Cancel gombra kattint.	A Register gomb után a felület jelzi, hogy nem töltötte ki a mezőket, majd a Cancel gombra kattintás után a modal eltűnik.
add-parkinghouse-modal spec	A Parkinghouses fülre navigál. Megvizsgálja, hogy megjelentek-e a szükséges elemek, majd megnyitja a parkolóház hozzáadói modalt. Majd először az Add gombra, majd a Cancel gombra kattint.	Az Add gomb után a felület jelzi, hogy nem töltötte ki a mezőket, majd a Cancel gombra kattintás után a modal eltűnik.
add-level-modal spec	A Parkinghouses fülre navigál. Megvizsgálja, hogy megjelentek-e a szükséges elemek, majd kiválasztja a kívánt parkolóházat és megnyitja a szint hozzáadói modalt. Majd először az Add gombra, majd a Cancel gombra kattint.	Az Add gomb után a felület jelzi, hogy nem töltötte ki a mezőket, majd a Cancel gombra kattintás után a modal eltűnik.

add-slot-modal spec	<p>A Slots fülre navigál. Megvizsgálja, hogy megjelentek-e a szükséges elemek. Kiválasztja a kívánt parkolóházat és annak szintjét. Megnyitja a parkolóhely hozzáadói modalt. Majd először az Add gombra, majd a Cancel gombra kattint.</p>	<p>Az Add gomb után a felület jelzi, hogy nem töltötte ki a mezőket, majd a Cancel gombra kattintás után a modal eltűnik.</p>
edit-slot-modal spec	<p>A Slots fülre navigál. Megvizsgálja, hogy megjelentek-e a szükséges elemek. Kiválasztja a kívánt parkolóházat és annak szintjét. Megnyitja a parkolóhely szerkesztői modalt. Majd először az Update gombra, majd a Delete gombra kattint.</p>	<p>Az Update gomb után a felület jelzi, hogy frissítette a kívánt helyeket, majd a Delete gombra kattintás után a helyek törlődnek.</p>

3.6.2. API végpontok tesztelése

A program tesztelés során az egységtesztekhez az XUnit[17] tesztelőkönyvtárat használja. Ez a könyvtár lehetővé teszi a C# programok egyszerű és gyors tesztelését. A tesztek "AAA" alapú tesztelési formát követnek (Arrange, Act, Assert). Mivel a legtöbb API végpont hasonló felépítésű és működésű metódusokból áll, a táblázatban nem minden teszteset szerepel.

3.2. táblázat. API végpontok tesztelése

Végpont	Teszteset	Elvárt eredmény
AddParkingHouse	AddParkinghouseSuccessful	Sikeres létrehozás
AddParkingHouse	AddParkinghouseFailed	Sikertelen létrehozás
AddParkingHouse	AddParkinghouseSuccessful CheckDatas	Sikeres létrehozás után megfelelő adatok
UpdateParkingHouse	UpdateParkinghouseSuccessful	Sikeres frissítés
UpdateParkingHouse	UpdateParkinghouseFailed	Sikertelen frissítés

GetAllParkingHouses	GetParkinghousesSuccessful	Sikeresen lekéri az összes parkolóházat
GetAllParkingHouses	GetParkinghousesSuccessful CheckDatas	Sikeress lekérdezés utáni adatok megfelelőek
GetAllParkingHouses	GetParkinghousesSuccessful ContainsMoreThanOneElement	Sikeress lekérdezés után megfelelő adatmennyiség
AddLevel	AddLevelSuccessful	Sikeress létrehozás
AddLevel	AddLevelFailed	Sikertelen létrehozás
AddLevel	AddLevelSuccessfulCheckDatas	Sikeress létrehozás után megfelelő adatok
UpdateLevel	UpdateLevelSuccessful	Sikeress frissítés
UpdateLevel	UpdateLevelFailed	Sikertelen frissítés
GetAllLevels	GetLevelsSuccessful	Sikeresen lekéri az összes parkolóházat
GetAllLevelsHouses	GetLevelsSuccessful CheckDatas	Sikeress lekérdezés utáni adatok megfelelőek
GetAllLevels	GetLevelsSuccessfulContains MoreThanOneElement	Sikeress lekérdezés után megfelelő adatmennyiség

4. fejezet

Összegzés

Az általam létrehozott Parkolóház foglalórendszer segítségének köszönhetően a parkolóhely keresése már a múlté, hiszen egy egyszerű, könnyedén kezelhető oldalon már előre lefoglalhatjuk a számunkra megfelelő parkolóhelyet a következő napra.

Az alkalmazás fejlett technológiákra épül, ezért bármilyen operációs rendszeren használható. A modern megjelenés pedig minden ablakméret konfigurációban kivá-
lóan átlátható, így bárki bárhol kényelmesen foglalhat, legyen szó számítógépről,
tabletről vagy akár mobiltelefonról.

4.0.1. Továbbfejlesztési lehetőségek

A program minősége több szempontból is továbbfejleszthető, más keretrendsze-
rek, illetve modernebb tervezési minták, programarchitektúrák felhasználásával.

Fizetés bevezetése

Az alkalmazásba könnyen beépíthető egy fizető rendszer, ahol havi vagy akár éves
tagsági díj befizetésére, annak ellenőrzésére lenne lehetőség. Az is megvalósítható,
hogy tagsági díj helyett minden parkolásnak egységes ára legyen és ezt számlázná
ki havonta vagy akár évente.

Irodalomjegyzék

- [1] *Docker*. <https://docs.docker.com/>.
- [2] *Angular Bootstrap*. <https://ng-bootstrap.github.io/#/home>.
- [3] *Node.js*. <https://nodejs.org/en>.
- [4] *Visual Studio Code*. <https://code.visualstudio.com/>.
- [5] *Git*. <https://git-scm.com/>.
- [6] *.Net*. <https://learn.microsoft.com/en-us/dotnet/>.
- [7] *Visual Studio Community 2022*. <https://visualstudio.microsoft.com/vs/community/>.
- [8] *Angular*. <https://angular.io/docs>.
- [9] *Github Desktop*. <https://desktop.github.com>.
- [10] *MSSQL*. <https://learn.microsoft.com/en-us/sql/?view=sql-server-ver16>.
- [11] *Draw.io*. <https://app.diagrams.net/>.
- [12] *Entity Framework Core*. <https://learn.microsoft.com/en-us/ef/>.
- [13] *Identity Framework*. <https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-7.0&tabs=visual-studio>.
- [14] *MVC architecture*. https://www.tutorialspoint.com/struts_2/basic_mvc_architecture.htm.
- [15] *Dependency Injection*. <https://www.tutorialspoint.com/explain-dependency-injection-in-chash>.
- [16] *Cypress*. <https://www.cypress.io>.

- [17] XUnit. <https://learn.microsoft.com/en-us/dotnet/core/testing/unit-testing-with-dotnet-test>.

Ábrák jegyzéke

2.1.	Bejelentkezési felület	7
2.2.	Bejelentkezési felület	8
2.3.	Regisztrációs felület	9
2.4.	Jelszóváltoztatási felület	10
2.5.	<i>Home</i> felület	11
2.6.	<i>Companies</i> felület	12
2.7.	<i>Users</i> felület	12
2.8.	<i>Parkinghouses</i> felület	13
2.9.	<i>Slots</i> felület	13
2.10.	<i>Reserve</i> felület	14
2.11.	<i>My reservations</i> felület	14
2.12.	Nem engedélyezett cég vagy felhasználó esete	15
2.13.	Eltávolítás	16
2.14.	Felhasználók felvétele	17
2.15.	Parkolóházak felvétele	18
2.16.	Szintek felvétele	19
2.17.	Parkolóhelyek felvétele	20
2.18.	Parkolóhelyek személyre szabása	21
2.19.	Parkolóhelyek személyre szabása	22
3.1.	<i>CompanyAdmin</i> felhasználói esetdiagram	25
3.2.	<i>Boss/Employee</i> felhasználói esetdiagram	25
3.3.	<i>SystemAdmin</i> felhasználói esetdiagram	26
3.4.	<i>Home</i> felületek drótvázterve	26
3.5.	<i>Companies</i> felület drótvázterve	27
3.6.	<i>Users</i> felület drótvázterve	27
3.7.	<i>Slots</i> felület drótvázterve	28
3.8.	Csomag diagram	32

3.9.	Adatbázis séma	33
3.10.	<i>UserProfile</i> osztálydiagramja	34
3.11.	<i>SystemAdmin</i> osztálydiagramja	34
3.12.	<i>CompanyAdmin</i> osztálydiagramja	35
3.13.	<i>DefaultUser</i> osztálydiagramja	35

Táblázatok jegyzéke

3.1. Felhasználói felület tesztelése	36
3.2. API végpontok tesztelése	38