



ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

**NEUMANN JÁNOS
INFORMATIKAI KAR**



SZAKDOLGOZAT

**OE-NIK
2022**

Hallgató neve:
Hallgató törzskönyvi száma:

**Bolyki Ágnes
T/006146/FI12904/N**

SZAKDOLGOZAT FELADATLAP

Hallgató neve: **Bolyki Ágnes**
Törzskönyvi száma: T/006146/FI12904/N
Neptun kódja: 

A dolgozat címe:

**Okos egyetemi parkoló hely monitorozó rendszer
Smart parking system for OU**

Intézményi konzulens: Lovas István
Külső konzulens:

Beadási határidő: 2021. december 15.

A záróvizsga tárgyai: Számítógép architektúrák
IoT, beágyazott rendszerek és robotika
specializáció

A feladat

Tervezzon és valósítson meg egy okos parkoló hely monitorozó rendszert az Óbudai Egyetem dolgozói részére. A cél egy olyan rendszer kialakítása, amely képes a parkolóhelyek nyilvántartására egy adatbázisban és online, interaktív alkalmazásban ezek megjelenítésére, lefoglalására. Az alkalmazásban a szabad parkolóhelyek közül lehessen választani, lefoglalni. A parkolás kezdetét és a terület elhagyását a helyszínen egy erre kialakított rendszer tartsa nyilván.

A dolgozatnak tartalmaznia kell:

- a megoldandó feladat pontos leírását és részletes elemzését,
- a meglévő rendszerek elemzését,
- az alkalmazni kívánt technológiák elemzését,
- rendszertervet, a döntések indoklásait,
- a megvalósítás leírását,
- a tesztelési tervet és a tesztek eredményeit.



.....
Dr. Fleiner Rita
intézetigazgató

A szakdolgozat elévülésének határideje: **2023. december 15.**
(OE TVSz 55.§ szerint)

A dolgozatot beadásra alkalmasnak tartom:

.....
külső konzulens

.....
intézményi konzulens



ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat / diplomamunka saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatomban / diplomamunkámban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja.

Budapest, 2022. 05. 12.

Bolfa Ágnes

hallgató aláírása



ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

KONZULTÁCIÓS NAPLÓ

Hallgató neve: Neptun kód: Tagozat:
Bolyki Ágnes.....
Telefon:

Szakedolgozat / Diplomamunka címe magyarul:

Okos egyetemi parkoló hely monitorozó rendszer.....

Szakedolgozat / Diplomamunka címe angolul:

Smart parking system for OU

Intézményi konzulens: Külső konzulens:

Lovas István

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

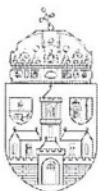
Alk.	Dátum	Tartalom	Aláírás
1.	2021.09.21	Témaválasztási konzultáció	
2.	2021.10.14	Szakirodalom áttekintése	
3.	2021.11.18	Lehetséges megvalósítás áttekintése	
4.	2021.12.08	Tartalmi, formai ellenőrzés	

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakedolgozat I. / Szakedolgozat II. (BSc) vagy Diplomamunka 1 / Diplomamunka 2 / Diplomamunka 3 / Diplomamunka 4 tantárgy követelményét teljesítette, beszámolóra / védésre bocsátható.

Budapest, 2021.12.09

Intézményi konzulens



KONZULTÁCIÓS NAPLÓ

Hallgató neve: Neptun kód: Tagozat:
Bolyki Ágnes Nappali
Telefon: Levelezési cím (pl: lakcím):

Szakdolgozat / Diplomamunka címe magyarul:

Okos egyetemi parkoló hely monitorozó rendszer

Szakdolgozat / Diplomamunka címe angolul:

Smart parking system for OU

Intézményi konzulens: Külső konzulens:

Lovas István.....

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2022.02.16	Specifikáció áttekintése	
2.	2022.03.09	Rendszerterv áttekintése	
3.	2022.04.06	Fejlesztés, tesztelés ellenőrzés	
4.	2022.05.04	Tartalmi, formai ellenőrzés	

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplomamunka 2 / Diplomamunka 3 / Diplomamunka 4 tantárgy követelményét teljesítette, beszámolóra / védésre bocsátható.

Budapest, 2022. 05. 11.

.....
Intézményi konzulens

Tartalomjegyzék

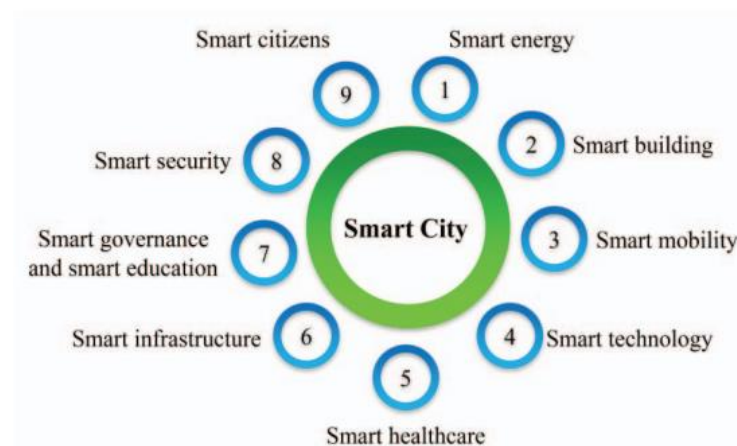
1.	Bevezetés.....	1
2.	Az okos parkolórendszerek kategóriái	3
2.1.	PGIS	3
2.2.	Tranzit alapú információs rendszer.....	3
2.3.	Okos fizetési rendszer	4
2.4.	E-parking.....	4
2.5.	Automatizált parkolás	4
2.6.	Összegzés	4
3.	A jelenleg használatban lévő rendszer működésének áttekintése	5
4.	A bevezetni tervezett rendszer folyamatainak ismertetése.....	6
5.	Jármű detektálási technológiák.....	7
5.1.	Mikrohullámú radar	7
5.2.	Passzív akusztikus érzékelők	8
5.3.	Passzív infravörös szenzorok	8
5.4.	RFID.....	8
5.5.	Ultrahangos szenzor	9
5.6.	Videó képfeldolgozás.....	9
5.7.	Összegzés	9
6.	Jármű azonosítási technológiák	11
6.1.	Az ANPR működése	12
6.2.	A piacon fellelhető termékek rendszámfelismerésre	12
6.2.1.	Plate recognizer	12
6.2.2.	Tesseract OCR.....	13
6.2.3.	Rekor's OpenALPR.....	13
6.2.4.	Összegzés	14
7.	Az alkalmazás fejlesztéséhez használatos keretrendszerek.....	15
7.1.	Architekturális minták webes alkalmazás fejlesztéséhez	15
7.1.1.	MVC	15
7.1.2.	MVVM	15
7.2.	Backend.....	16

7.3. Frontend	16
8. Rendszerterv	17
9. Követelmény specifikáció	18
9.1. Felhasználók jogkörei	18
9.2. Regisztráció	19
9.3. Belépés	20
9.4. Foglалás létrehozásának menete	20
9.5. Adminisztrátorok feladatköre.....	20
9.6. A parkoló területén működő rendszerrel szemben támasztott elvárások	20
10. A rendszer fejlesztéséhez választott eszközök ismertetése.....	22
10.1. Adatbázis	22
10.2. Backend	23
10.3. Frontend.....	24
10.4. Beágyazott rendszer.....	27
11. A rendszer felépítése.....	29
12. Megvalósítás	30
12.1. Adatbázis	30
12.2. Backend	33
12.2.1. AuthController	33
12.2.2. ReservationController	34
12.2.3. ParkingLotController	35
12.2.4. Services	37
12.3. Frontend.....	37
12.4. A parkoló területén működő rendszer	43
12.4.1. Kamerás rendszer.....	43
12.4.2. Parkolóhelyek foglaltsági állapotát jelző rendszer	45
12.4.3. A parkoló területén lévő tájékoztató rendszer	47
13. Tesztelés.....	51
13.1. API tesztek.....	51
13.2. Kliens tesztek.....	52
13.3. ANPR tesztek	53

13.4.	Jelenlét érzékelés tesztek	55
14.	Továbbfejlesztési lehetőségek	59
15.	Összefoglaló.....	60
16.	Summary	61
17.	Irodalomjegyzék	62
18.	Ábrajegyzék	65
19.	Táblázatjegyzék	67

1. BEVEZETÉS

Szakedolgozatom célja egy okos parkoló rendszer fejlesztése, mely alkalmazásának segítségével az autósok képesek egyszerűen, számukra szimpatikus parkoló helyet lefoglalni, a foglalást megérkezésig, vagy legalábbis reális időintervallumon belül fenntartani. Az ilyen és egyéb okos rendszerek napjainkban egyre nagyobb teret nyernek az emberek életének megkönnyítése, egyszerűbbé tétele, segítése érdekében. Az IoT rendszerek folyamatos fejlődésével, elterjedésével már mindennapjaink részévé váltak. Kezdeményezések indultak el az okos városok kialakítására, melynek szerves részét képezik az okos parkoló rendszerek, melyeket a városok nagy, és sokak véleménye szerint lassan élehetlenné váló közúti forgalom enyhítésére hoztak létre.



1.1. ábra: Az okos városok alkotó egységei [1]

A közlekedési csomópontokban feltorlódott forgalom miatt nehézkes az eljutás egyik helyről a másikra, az úticélunkhoz közeli parkolóhely találása a megtelt parkolóban. Ezek a torlódások súlyos biztonsági problémát jelentenek, balesetekhez vezethetnek. Az okos parkolóhelyek létrehozásának célja az ilyen jellegű balesetek elkerülése, illetve, hogy az autósok minél hamarabb találjanak parkolóhelyet, ezáltal a frusztráció érzésének csökkentése és a torlódások megakadályozása. Továbbá a minél kevesebb idő alatt történő megfelelő parkolóhely megtalálása, így csökkentve a légszennyezés mértékét.

A városok népességének folyamatos növekedése miatt az infrastruktúra és a szolgáltatások fejlesztése elengedhetetlen ahhoz, hogy a városok továbbra is funkcionálisak maradjanak. Ez hozzájárult a különféle digitális eszközök nagy mértékű térnyeréséhez, melyek egymással az interneten kommunikálva alkotják az IoT-t. Az IoT mint fogalom, azt jelenti, hogy különféle eszközök, rendszerek egymással Interneten keresztül kommunikálnak, osztanak meg és továbbítanak adatokat. „Thing” tulajdonképpen bármilyen objektum lehet, amihez hozzárendelhető egyedi azonosító és képes adatok továbbítására.

A parkolóház üzemeltetők számára hasznos információk nyerhetők ki a rendszerből, adatok elemzése útján tudják a parkolóház jövőbeni kihasználtságát előre jelezni, profitmaximalizálás céljából szükséges intézkedéseket bevezetni.

A felhasználók számára nyújtott információk alapján pedig biztosítva van annak a lehetősége, hogy elkerüljék a megtelt parkolókat, így nem követik el a hibát, ami ilyen rendszerek nélkül történne, hogy autójukkal behajtanak egy parkolóba, ott körözve üres helyet próbálnak keresni, miközben idejüket vesztegetik és üzemanyag fogyasztásuk nő.

A szakdolgozatomban elvárás egy olyan rendszer létrehozása, melynek segítségével a felhasználók egy alkalmazáson keresztül képesek parkolóhelyet foglalni, a parkoló bejáratánál kamerás rendszer azonosítja az érkező autókat, a regisztrált járművek, illetve az alkalmazásban helyet foglalt autók számára belépést biztosít. A parkoló területén beágyazott rendszer biztosítja a helyek foglaltságának visszajelzését, érzékeli, hogy az adott helyen áll-e éppen autó, a lefoglalt helyeken piros, a szabad helyeken pedig zöld fényt ad tájékoztatást.

Célom az okos parkolórendszerek néhány kategóriájának ismertetése, a jelenlegi rendszer működésének áttekintése, az újonnan bevezetni kívánt parkolórendszer folyamatainak bemutatása. Ezt követően a szakdolgozatom szempontjából lehetséges megvalósításokat három fő logikai egységre osztva ismertetem: az érkező autó azonosítására szolgáló technológiákat, a beléptetést követően annak ellenőrzését, hogy az autós az általa kiválasztott helyet elfoglalta-e, illetve a foglalásokat kezelő alkalmazást.

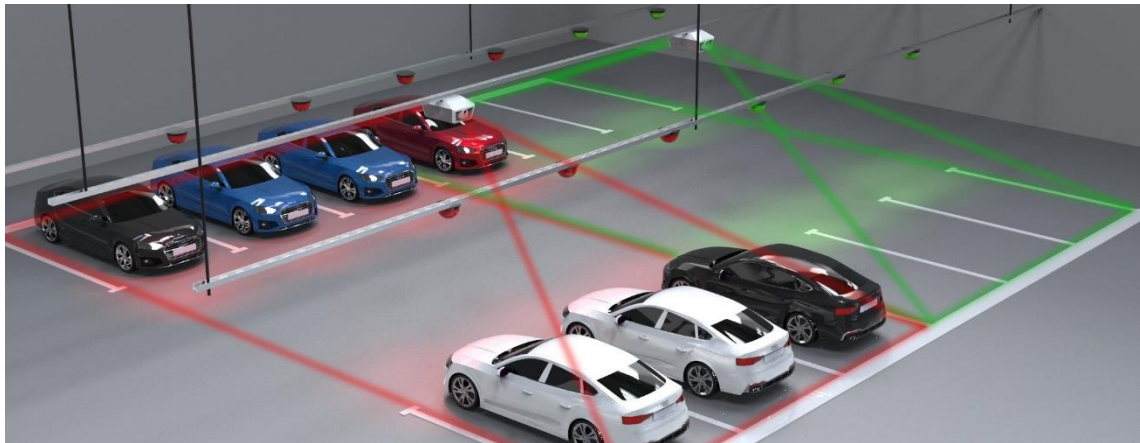
2. AZ OKOS PARKOLÓRENDSZEREK KATEGÓRIÁI

A [2] forrás szerint az okos parkolórendszereket öt kategóriába sorolhatjuk, ezek:

- parkolási útmutatás és információs rendszer (parking guidance and information system - PGIS),
- tranzit alapú információs rendszer,
- okos fizetési rendszer,
- E-parking,
- automatizált parkolás.

2.1.PGIS

Két fő kategóriát különböztetünk meg: egész város területére kiterjedő, és egy bizonyos parkolóházon vagy létesítményen belüli. Mindkettő segíti a felhasználót abban, hogy parkolóhelyet keressenek az úticéljukhoz webes felületen. [2] Az autós jelenlegi pozíciója meghatározható GPS használatával egy mobileszköz segítségével. Fő célja a rendszernek, hogy a parkolóhely keresésével töltött időt csökkentse, ugyanis ez néhány esetben nagyban növelheti az utazással töltött időt. Dinamikus információt ad a vezetőnek arról, hogy melyik területen hol találhatók az üres parkolóhelyek. A legtöbb ilyen rendszer lehetővé teszi az üres helyek megtalálását és a saját jármű megtalálását a parkoló területén. [3]



2.1. ábra: Beltéri PGIS

2.2.Tranzit alapú információs rendszer

PGIS-hez hasonló rendszer, annyi különbséggel, hogy ez park-and-ride lehetőséget biztosít, azaz a parkolók telítettsége és a tömegközlekedési menetrendek alapján keres olyan parkolót, ahol a felhasználó le tudja tenni járművét, lehetőleg minél közelebb az úti cél felé induló tömegközlekedési eszköz megállójához, hogy könnyen át lehessen szállni erre az eszközre. Ez a rendszer hozzájárul a tömegközlekedési szolgáltató bevételeinek növekedéséhez.

2.3.Okos fizetési rendszer

A hagyományos fizetési metódusok korlátjai miatt új fizetési technológiák bevezetésére van szükség. Korlátot jelentenek a hagyományos, készpénzzel történő fizetéseknél a késleltetések, mert sok idő amíg az autósok találják a parkoló automatát, és mire készpénzben kiszámolják a fizetendő összeget. Lényegében ezt a hagyományos fizetési technikát szeretnék kiváltani kontakt és kontakt nélküli fizetéssel, ezek történhetnek kontakt fizetés esetén okos-hitel-bank kártyával, kontakt nélküli fizetésnél kontakt nélküli kártyával és mobilos fizetéssel, de a kontakt metódus is megkívánja a parkolóautomata használatát, ezért a kontakt nélküli fizetés használatára próbálnak törekedni, azonban ezzel szemben az emberek szkeptikusak, jogosan féltik a személyes és számla adataikat.

2.4.E-parking

Lehetőséget biztosít, hogy a felhasználók a parkolóhelyek foglaltságának állapotát lekérdezzék, helyet foglaljanak, ezzel biztosítva, hogy a hely üres legyen addig, amíg meg nem érkeznek. Ezek a rendszerek többféle platformról is hozzáférhetők. Továbbá könnyen bővíthető fizetési lehetőséggel, beágyazhatók az okos fizetési rendszerek funkciói.

2.5.Automatizált parkolás

Számítógép által vezérelt mechanizmust vesz igénybe, az autónak csak annyi teendője van, hogy egy meghatározott helyre vezesse autóját, bezárja azt, és innentől a mechanizmus eljuttatja a parkolóba. Ennek a megoldásnak az előnye, hogy a helyek megfelelően ki vannak használva, nem úgy, mint a hagyományos parkolók esetén, ahol az utat biztosítani kell, hogy autóval el lehessen menni a parkolóhelyre. Olyan területeken érdemes az implementálása, ahol a kialakításból adódóan kevés hely van. Előnyei közé tartozik az is, hogy a vezetőnek nem kell bemennie a parkolóba, mivel automatikusan történik az autó elhelyezése, ez pedig biztonságos mind a vezetőre, mind az autóra nézve, mert elkerülik a szűk, zsúfolt helyeken a vezetést.

2.6.Összegzés

Szakdolgozatomban E-parking rendszert szeretnék megvalósítani, okos fizetési funkció használata nélkül, mivel ez a megoldás valószínűleg leginkább az elvárt működést, ugyanis fontos, hogy legyen lehetőség a parkolóhelyek előzetes foglalására, illetve a rendszer kiépítése nem egy konkrét parkolóban valósulna meg, ezért nem szükséges a PGIS és a tranzit alapú információs rendszerek funkcióit alkalmazni, továbbá nem elvárás az automatizált parkoló kiépítése sem.

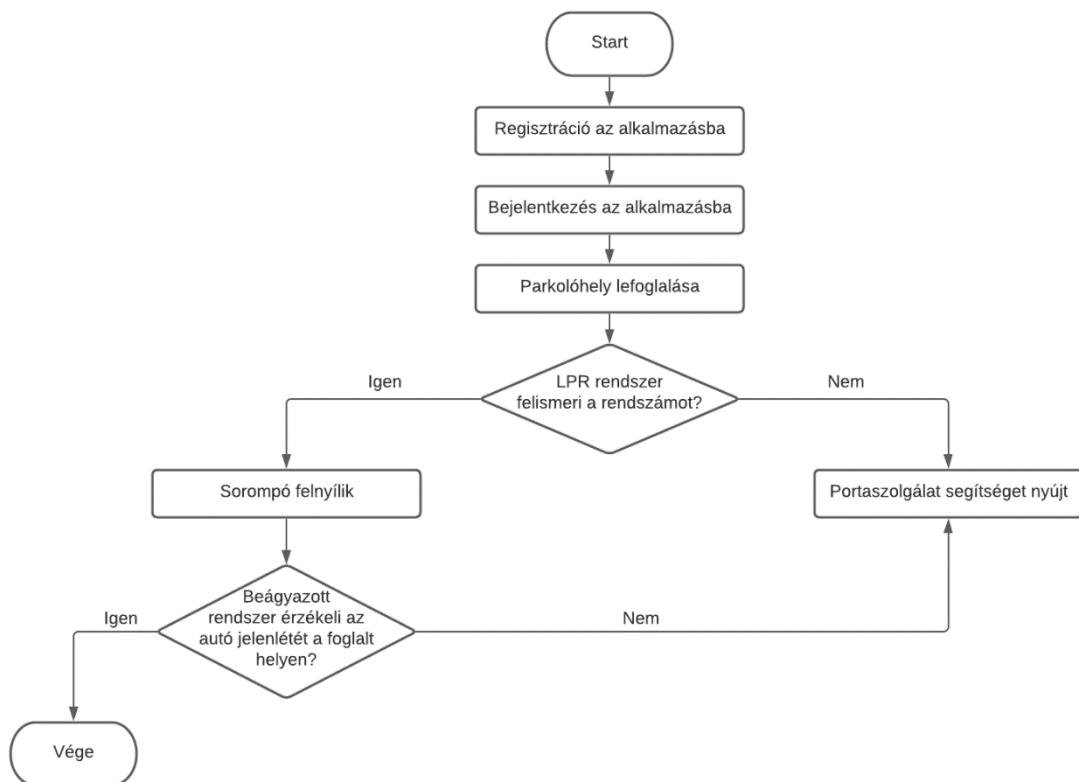
3. A JELENLEG HASZNÁLATBAN LÉVŐ RENDSZER MŰKÖDÉSÉNEK ÁTTEKINTÉSE

Jelenleg a parkoló bejáratánál az érkező dolgozók belépőkártyájukkal nyitják a sorompót, ha ez esetleg nem működne, a portaszolgálaton kérhetnek segítséget. A parkolón belül vannak fenntartott helyek bizonyos személyek számára, viszont ezek nincsenek mindig kihasználva.

A használatban lévő rendszer jól működik, de van lehetőség a továbbfejlesztésére egy olyan rendszer kiépítésével, amely lehetővé teszi, hogy a dolgozók előre tudják jelezni egy webes felületen, hogy mikor tartanak igényt a parkolóhelyre. Ezáltal elkerülhető a parkoló túlszűfolttsága, mert érkezés előtt meg lehet nézni ennek telítettségét. Ugyanakkor fontos, hogy ha valaki elfelejtene foglalni, de még vannak szabad helyek, akkor be tudjon jutni a parkolóba. Ennek jelzésére célszerű lenne egy kijelzőn mutatni a bejáratnál a szabad helyek számát, így, ha már foglalt az összes hely, akkor az érkezők be se mennének.

4. A BEVEZETNI TERVEZETT RENDSZER FOLYAMATAINAK ISMERTETÉSE

Az alap koncepció szerint a parkolóba érkezés előtt az autósok lefoglalják online kommunikációra képes eszközükön a számukra szimpatikus helyet a regisztrált parkolón belül. Érkezésükkor a parkoló bejáratánál a zárt sorompó előtt rendszámfelismerő rendszer működik, amely ellenőrzi az autó rendszámát és ha van aktív foglalása akkor beengedi a sorompó, ellenkező esetben a bejáratnál elhelyezett kijelzőn kaphat tájékoztatást, hogy mennyi a szabad helyek száma és ez alapján dönthet úgy, hogy máshol keres szabad helyet, vagy a még szabad helyek egyikét elfoglalja, ekkor a hagyományos kártyás módszerrel léphet be. A belépést követően az alkalmazásban lefoglalt helyre szükséges parkolnia az autónak, hogy ne foglalja el mások helyét, megtartva ezzel a rendet. Amíg az autó be nem áll a helyére, addig ott egy visszajelző LED folyamatosan villog pirosan, a parkolást követően pedig kikapcsol. Ha egy hely szabad, akkor ez a LED zölden világít, de ha a belépő autó elfoglalja ezt a helyet, akkor jelzést küldjön, hogy rossz helyen áll. A parkoló elhagyásával a felszabadult hely újra foglalásra elérhető lesz az alkalmazásban.



4.1. ábra: A tervezett rendszer főbb folyamatainak ismertetése

5. JÁRMŰ DETEKTÁLÁSI TECHNOLOGIÁK

Fontos feladat az okos parkolórendszereknél a foglaltság monitorozásának megszervezése, ugyanis ez ad információt az autósok és az üzemeltetők számára is. Rengeteg féle szenzor létezik ennek a feladatnak a megvalósítására. A járművek detektálására használatos technológiák kiválasztásakor szem előtt kell tartani a parkoló méretét és kialakítását. A járművek detektálására alkalmazható szenzorok és rendszerek két fő csoportba sorolhatók intruzív és nem intruzív. Az intruzív szenzorok elhelyezéséhez az út felületének megbontására van szükség, ezzel szemben a másik csoportba tartozó szenzorok elhelyezéséhez nem szükséges semmilyen beavatkozás, azok könnyen elhelyezhetők. A két csoporthoz tartozó szenzorok és technológiák közül néhányat tartalmaz az alábbi táblázat: [2]

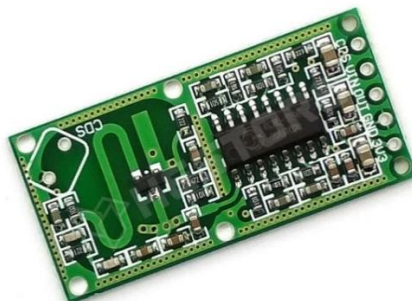
Intruzív	Nem intruzív
•aktív infravörös szenzor •induktív hurokdetektor •magnetométer •magnetorezisztív érzékelő •pneumatikus érzékelők •piezoelektromos érzékelő •WIM rendszer (weigh-in-motion)	•mikrohullámú radar •passzív akusztikus érzékelő •passzív infravörös szenzor •RFID •ultrahangos érzékelő •videó képfeldolgozás

5.1. táblázat: Példák intruzív és nem intruzív szenzorokra

A megtervezni kívánt rendszer szempontjából célszerű nem intruzív technológiát választani, ugyanis ennek üzembe helyezése nem igényel különösebb előkészületeket, nem szükséges a környezet megváltoztatása, ezáltal ára is kedvezőbb.

5.1.Mikrohullámú radar

Mozgásérzékelésre alkalmas, mikrohullámot bocsát ki és ezek verődnek vissza a környezetben lévő tárgyakról, ha az érzékelőhöz viszonyítva valami mozog, akkor ez a visszaverődés megváltozik, más lesz a hullám frekvenciája és hullámhossza, ezt nevezzük Doppler-effektusnak. Érzéketlenek az időjárás változásra, képes több forgalmi sáv adatainak gyűjtésére és a jármű sebességének közvetlen mérésére. [2] [4]



5.1. ábra: Mikrohullámú radaros mozgásérzékelő modul

5.2.Passzív akusztikus érzékelők

Mikrofonnal vannak felszerelve, és tartalmazzák az áramkört, amely a mikrofon jeleit elemzi, a beállításoktól függően bizonyos zajszint felett működésbe lépnek, és elvégzik a definiált feladatot. Érzékeny a hideg időre, bizonyos típusai nem ajánlottak lassan mozgó járművek detektálására. Előnyei közé tartozik, hogy több forgalmi sávot képes egyszerre megfigyelni és érzéketlen a csapadékra. [2]



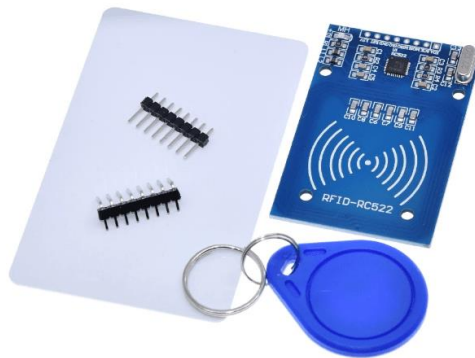
5.2. ábra: Digitális akusztikus érzékelő

5.3.Passzív infravörös szenzorok

Az élőlények, tárgyak által kibocsátott infravörös sugárzás érzékelésére képesek, az eszköz nem bocsát ki sugárzást. A mozgást csak közvetve tudják detektálni. Előnyt jelent, hogy az egy területen elhelyezett több infravörös érzékelő nem befolyásolja egymás működését. Hátránya pedig az, hogy eső, hó és köd esetén nem megbízható. [5]

5.4.RFID

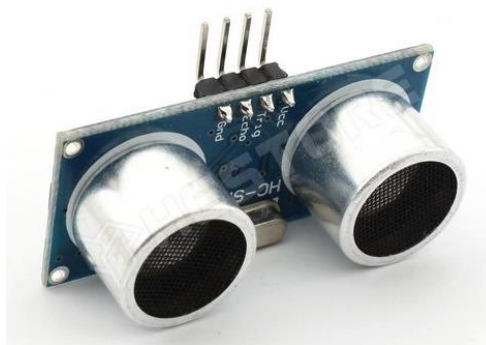
Olvasóból és tagból áll, az olvasó képes leolvasni a közelében található tag-ek megfelelő memóriaterületét, majd az innen kinyert adatokat további feldolgozásra elküldi az erre alkalmas egységnek. Megkülönböztetünk passzív, fél passzív és aktív tageket, a passzív tag-ek egy antennából és chipből állnak, a fél passzív és aktívak pedig tartalmaznak energiaforrást is. Mindegyik tagnek van egyedi azonosítója, ami általában nem változtatható. Ennek a technológiának előnye, hogy viszonylag költséghatékony, viszont hátránya, hogy minden autóra fel kell rögzíteni. [6]



5.3. ábra: RFID kártya író és olvasó modul

5.5. Ultrahangos szenzor

Feszültség hatására az eszköz ultrahang tartományú hanghullámot enged ki hangszóróján, ami visszaverődik a közelben lévő tárgyakról az eszköz mikrofonjába, ezt az erősítő áramkör feszültség impulzussá alakítja, amit a mikrovezérlő fel tud dolgozni. [8]



5.4. ábra: Ultrahangos távolságérzékelő modul

5.6. Videó képfeldolgozás

Szükséges hozzá: egy vagy több kamera, szoftver a képek értelmezéséhez, és mikroprocesszoros számítógép a digitalizáláshoz és feldolgozáshoz. A képkockák elemzése alapján megállapítható a jármű mozgása, ugyanis a képkockák közötti különbségek erre engednek következtetni. A működéshez elengedhetetlen a megfelelő világítás, a működést befolyásoló tényezők közé tartozik például az árnyékok, a kamera lencséjén felhalmozódó szennyeződés. [2]

5.7.Összegzés

Az ismertetett technológiák közül az ultrahangos szenzort találtam a legmegfelelőbbnek, leghatékonyabbnak. Az RFID hátránya, hogy a megfelelő működéshez elengedhetetlen,

hogy az autókra felszereljük a tageket, ami többletmunkával járna és nem is lenne hatékony, azt is számításba kell venni, hogy mi történne, ha nem mindig egy állandó felhasználói kör venné igénybe a parkolóhelyeket. A passzív akusztikus érzékelők hátránya az árukban nyilvánul meg.

6. JÁRMŰ AZONOSÍTÁSI TECHNOLÓGIÁK

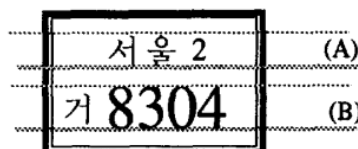
Az előző fejezetben áttekintő elemzést adtam a jármű detektálási technológiákról, kiválasztottam a számomra megfelelőt, amely a parkoló helyeken fogja a járművek jelenlétét érzékelni. Erre azért van szükség, hogy meg tudjuk figyelni, hogy a parkoló területére belépő autó a megfelelő, előre lefoglalt helyére áll-e. Azonban vannak a parkolóban olyan helyek, ahova csak bizonyos rendszámmal rendelkező autó állhat. Szükség van olyan technológiára, amely alkalmazásával ezeket az autókat azonosítani tudjuk. Az autókat célszerű rendszámuk alapján, valamilyen képfelismerési technológiával azonosítani. Ebben a fejezetben ezeket veszem sorra.

A [8] forrásban megjelölt kutatás szerint a rendszám felismerése három fő részből áll: jármű érzékelése, kép készítése, és maga a felismerés folyamata. Az érzékelés valamilyen szenzorral történik, ha ez érzékeli a járművet, akkor jelet küld, hogy le kell fényképezni, erre olyan fényképezőgépet célszerű használni, ami megfelelő zársebességgel rendelkezik, vagy olyan kamerát, ami megfelelő FPS-el működik, ezáltal nem lesz elmosódott a kép, ezt követően az elkészített kép feldolgozásra kerül, először is meg kell határozni, hogy a képen hol helyezkedik el a rendszámtábla, ezt követően az ezen található karaktereket kell azonosítani.

Ebben a kutatásban [8] először is a kamera képet kell megkapni, és valós időben feldolgozni, hogy észlelhető-e valamilyen változás, ha változás történt, akkor egy jármű belépett a kamera látóterébe, majd innen a mozgását végig követik, amíg az autó rendszáma nincs megfelelően közel a kamerához, hogy az jó minőségű, további feldolgozás szempontjából használható képet adjon. Számos szűrési lehetőség létezik: térbeli, időbeli, ezek kombinációja, az idő-, és térbeli és adaptív. Ezek közül a térbeli szűrés bizonyul a leghatékonyabbnak mozgó objektumok detektálására.

A rendszám helyének megállapítására használt módszer alapját az képezi, hogy az elkészített képen a rendszámon szereplő számok és a rendszámtábla háttere olyan kontrasztot képez, ami a képen máshol nem található. Az ebben a kutatásban választott módszer két részből áll: a rendszám kiválasztása a képen, majd ennek megerősítése.

Ezt a képfelismerési technológiát koreai rendszámokon tesztelték, melyek 8 karaktert tartalmaznak. A rendszámot két részre osztották, ez a felső és az alsó sor, így 3-5 karakter található a két sorban, az alsó sorban lévő karaktersorozatot próbálja megtalálni a rendszer, és ez alapján behatárolni a rendszámtábla helyét.



6.1. ábra: Koreai rendszám

Általánosságban elmondható, hogy a járművek rendszámáról készült kép feldolgozása OCR technológiával történik, így alakítva a képi információt digitális karakterekké. A rendszámtáblák országonként vagy államonként eltérnek egymástól, a karakterek nem néznek ki ugyanúgy, ezért fontos, hogy ha több ország vagy állam rendszámán is szeretnénk az algoritmust alkalmazni, akkor ezeknek a mintáknak a betanítása szükséges. A képfeldolgozás a rendszámok felismerésére és a jármű detektálására is használható. [9]

A rendszámok digitális információvá válásával több adat is rendelkezésünkre fog állni, mint például az adott autó mikor, milyen irányban, milyen sebességgel haladt át a kamera előtt, a jármű paraméterei, esetleg a sofőr adatai. [9]

6.1.Az ANPR működése

A rendszámfelismerés négy lépcsős folyamat. Az első két lépés a szokványos OCR folyamat: lokalizáció és szegmentáció. Ebben a két lépésben a kép egészén meg kell állapítani, hogy hol helyezkedik el a rendszámtábla, ezen kívül a környezeti információkat figyelmen kívül kell hagyni, majd az egyes karaktereket tartalmazó zónákat kell szétválasztani egymástól, hogy a karakterek egyesével legyenek azonosítva. A harmadik lépésben történik a karakterek felismerése zónánként, ezt követően pedig regionális szintaktikai korrekciót szükséges végezni, hogy az egyformának tűnő karakterek egyértelműen meg legyenek különböztetve egymástól (ilyen például a 0 és az o). [9]

6.2.A piacon fellelhető termékek rendszámfelismerésre

6.2.1. Plate recognizer

Két termék közül lehet választani: Snapshot és Stream. A rendszer működése a valós környezethez van tervezve, sötéthez, homályos képekhez, gyors autókhoz. A neurális hálózat a rendszámokra fókuszál, figyelmen kívül hagyva az autókön található matricákat. Több, mint 90 országban használható. A Snapshot egy képet fogad, ezen felismeri a rendszámot, ez alapján meg tudja határozni a régiót, a jármű típusát, modellét, színét, orientációját, és a haladás irányát. A Stream valós idejű kamera képről nyeri ki ugyanezeket az információkat. Az eredményeket json vagy csv formátumban adja vissza, webhooks-al vagy parkpow-al küldi el a saját rendszerünkbe. Az eredményeket tovább lehet küldeni az ALPR dashboardra, ami több funkcióval is rendelkezik. [10]

Parameter	Snapshot	Stream
Processes	Images	Live Camara, Video Files
Runs On	Cloud, On Premise	On Premise
Internet	Not Needed	Not Needed
Integration	API Code in 8 languages	Connect to camera, programmatically process videos
Hardware	Windows, Linux, Jetson, Pi, Kubernetes, etc.	Windows, Linux, Mac, Jetson
Speed	Cloud: 200 ms @ 99.9% SLA SDK: 50-100 ms on Midrange PC	6-10 Cameras on Midrange PC

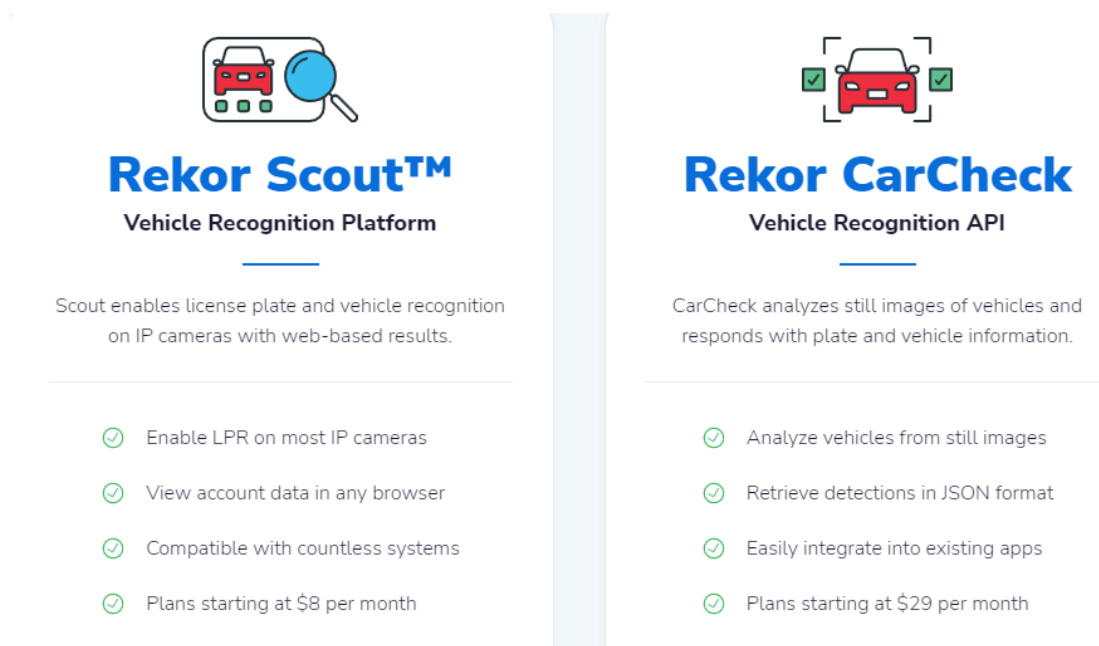
6.2. ábra: Snapshot és Stream összehasonlítása

6.2.2. Tesseract OCR

Népszerű nyílt forráskódú OCR motor. Közvetlenül, vagy API-ként használható a nyomtatott szöveg kinyerésére képekről. Több programozási nyelvvel és keretrendszerrel is kompatibilis. OCR mesterséges intelligenciát használ a szöveg keresésére és felismerésére a képeken. Mintákat keres a pixeleken, betűkben, szavakban és mondatokban. Kétféle megközelítést alkalmaz. Először is fel kell ismernie a karaktereket, majd a hiányzó betűket, melyek nem voltak megfelelően megadva, a mondat vagy szövegkörnyezetnek megfelelően kitölti. [11]

6.2.3. Rekor's OpenALPR

Szintén két termék közül lehet választani járműfelismerésre: Scout és CarCheck. A Scout szinte bármilyen IP, forgalmi vagy biztonsági kamerán lehetővé teszi az automatikus rendszám és jármű felismerést. Az eredményeket web alapú interfészen jeleníti meg. A CarCheck pedig egy API, mely képeket elemel. Az API hívásokhoz szinte bármilyen programozási nyelv használható. Mindkét megoldás havi előfizetéses. [12]



6.3. ábra: Scout és CarCheck összehasonlítása

6.2.4. Összegzés

A parkolórendszer megvalósításához a Plate Recognizer Snapshot termékét választom, mert kész rendszer, nem igényel nagyobb fejlesztést, emellett havonta 2500 keresésig ingyenes. Célszerű az on-premise megvalósítás, mert a cloud-dal ellentétben nincs meghatározva a képfájl mérete, ami cloud esetén 3MB, illetve nem igényel internet kapcsolatot.

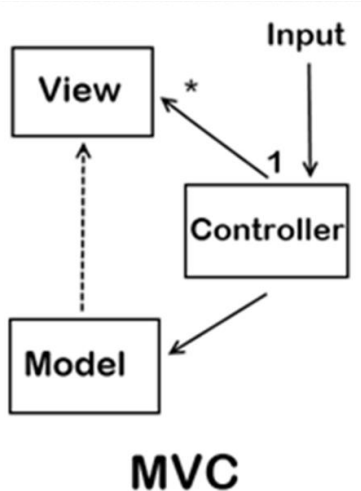
7. AZ ALKALMAZÁS FEJLESZTÉSÉHEZ HASZNÁLATOS KERETRENDSZEREK

A parkolóhelyek legfoglalása alkalmazáson keresztül fog történni. A felhasználónak regisztrálnia kell, majd belépést követően érhetőek el az alkalmazás funkciói. Célszerű erre egy webes alkalmazás fejlesztése, ugyanis ez minden internetképes eszközről böngésző segítségével hozzáférhető, ami lehetővé teszi, hogy bármikor és bárhol le lehessen helyet foglalni.

7.1. Architektúrális minták webes alkalmazás fejlesztéséhez

7.1.1. MVC

Három komponensből áll: Model-View-Controller. A felhasználói kérések a controllerhez érkeznek be, a controller a modellel dolgozik együtt, hogy a felhasználói kérésnek eleget téve adatokat adjon vissza. A controller feladata a nézetek közti váltás, a nézetek feltöltése a model adataival. Az MVC alkalmazásokban a model képviseli az alkalmazás jelenlegi állapotát és az általa végrehajtandó üzleti logikát és műveleteket. A view feladata a tartalom megjelenítése a felhasználónak a felhasználói interfészen keresztül, logikát csak úgy tartalmazhat, ha ez a megjelenítés szempontjából elengedhetetlen. A controllerek a felhasználó interakciókat kezelik, dolgoznak a modellel, és kiválasztják a megjeleníteni kívánt view-t. [13]

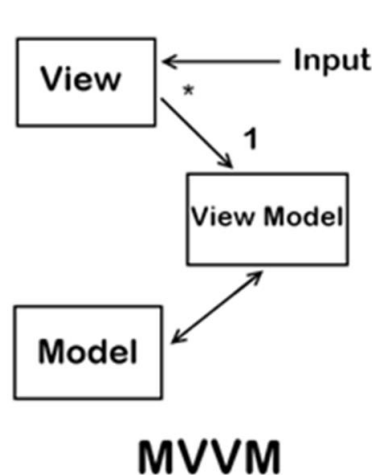


7.1. ábra: MVC architektúrális minta [15]

7.1.2. MVVM

Három komponense a model, view és viewmodel. A view ismeri a viewmodelt, és a viewmodel ismeri a modelt, de a model nem ismeri a viewmodelt és a viewmodel sem a viewt, ezáltal a viewmodel elválasztja a viewt a modeltől. A view meghatározza, hogy a felhasználó mit lát a képernyőn, nem használ üzleti logikát. A viewmodel olyan tulajdonságokat és parancsokat valósít meg, amelyekhez a nézet az adatokat köti, és a viewmodel a viewt eseményeken keresztül értesíti az állapotváltozásokról. Ezek a

tulajdonságok és parancsok meghatározzák a felhasználói interfész által kínált funkcionalitást, de a view dönti el, hogyan legyen ez a funkcionalitás megjelenítve. A feladatai közé tartozik még a view interakciók megvalósítása a model osztályok segítségével adat átalakítást használva, ahol szükséges. A model osztályok nem vizuális osztályok, melyek az alkalmazás adatait foglalják magukba, általában adatmodellt, illetve üzleti és validációs logikát tartalmaznak. [14]



7.2. ábra: MVVM architektúrális minta [15]

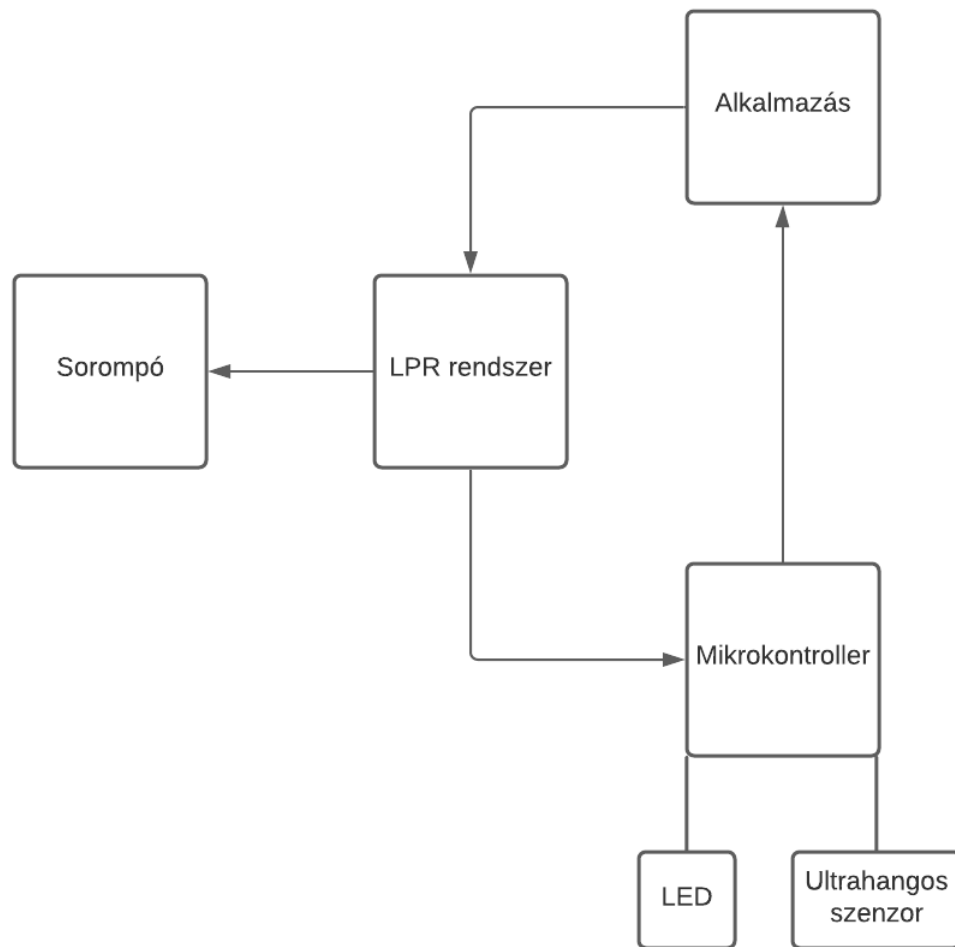
7.2. Backend

Az ASP.NET keretrendszert választom, amely egy Microsoft által fejlesztett keretrendszer, segítségével platformfüggetlen alkalmazásokat lehet fejleszteni .NET-el és C# nyelven. Az ASP.NET Web API az ASP.NET-re épülve támogatja a kérés-válasz alapú kommunikációt HTTP protokollon keresztül. Ideális választás RESTful alkalmazások fejlesztéséhez. [16]

7.3. Frontend

A frontend fejlesztésére az Angular keretrendszert választom. Előnyei közé tartozik, hogy alkalmas web, asztali és mobil alkalmazások fejlesztésére Typescript programozási nyelven. Támogatja a single page alkalmazásokat, melynek jellemzője, hogy egyetlen HTML oldalt tölt be, és dinamikusan frissíti az oldal tartalmát a felhasználói interakció alapján az egész oldal frissítése nélkül. A kétirányú adatkötésnek köszönhetően az adatok frissülése során a felhasználói felület is azonnal frissül. [17]

8. RENDSZERTERV



8.1. ábra: Saját rendszer terve

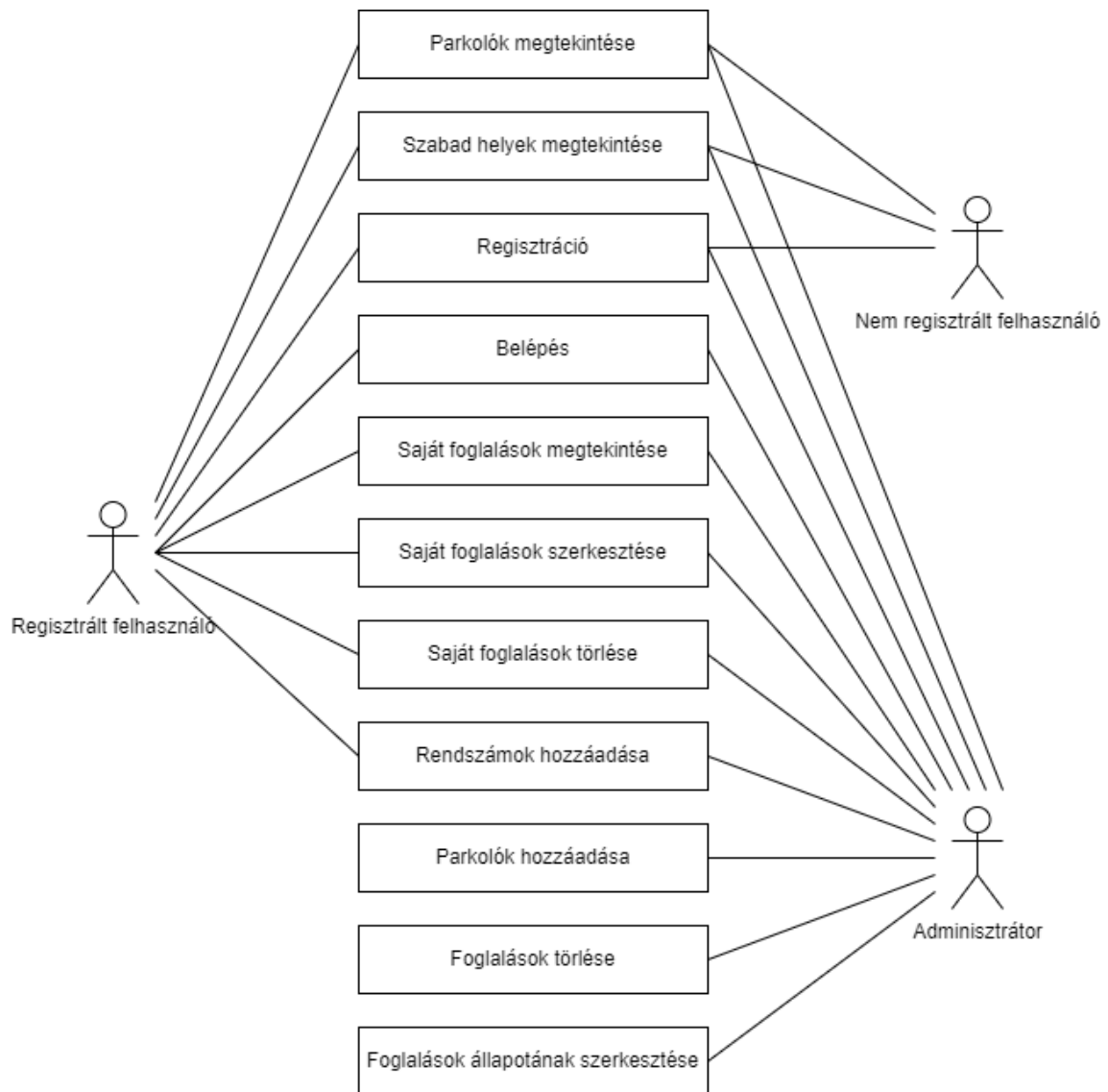
A felhasználók érkezésekor az LPR rendszer a rendszám alapján a sorompó felnyitásával engedheti be az autót, viszont, ha az adott időintervallumban nincs foglalása, akkor nem nyitja fel a sorompót. Mikor az autós a parkoló területére belépett, akkor a mikrokontroller által vezérelt LED folyamatosan villog, amíg be nem áll a helyére, a jelenlétét az ultrahangos szenzor érzékeli.

9. KÖVETELMÉNY SPECIFIKÁCIÓ

Ebben a fejezetben a rendszer működéséhez szükséges követelményeket ismertetem a felhasználók szempontjából. Az új rendszer célja a meglévő, hagyományos parkolási folyamat áttekinthetőbbé tétele, a felhasználók segítése parkolóhelyük kiválasztásában, az érkező és távozó autók nyilvántartása, ezáltal a parkoló optimális kihasználtságának támogatása. Fontos, hogy az új rendszer a jelenlegi parkolási folyamatot ne váltsa ki maximálisan, az maradjon továbbra is elérhető, így az alkalmazást használni nem szándékozó személyek is igénybe vehessék a parkolót, olyan módon, ahogy azt eddig is tették, de az alkalmazást használók számára adjon többlet információt a parkoló foglaltsági állapotáról, tegye lehetővé, hogy legyen számukra hely, mikor megérkeznek.

9.1.Felhasználók jogkörei

A rendszerben szükséges tárolni a fenntartott hellyel rendelkező autók rendszámát, számukra vannak helyek a parkolóban, amit mások nem vehetnek igénybe, az ő helyük az alkalmazásba történő regisztráció és foglalás nélkül is elérhető kell, hogy legyen. A parkoló területére történő belépésüket a kamerás rendszer rendszámfelismerő funkciója teszi lehetővé, ha a rendszer felismeri az érkező autó rendszámát és ez a tárolt rendszámmal megegyezik, akkor aktív foglalás nélkül is beengedi az érkező autót, ha valamilyen oknál fogva nem működne az elvárt módon a rendszámfelismerés, akkor a hagyományos módszerrel, belépő kártyájukkal nyithatják a kaput. A nem különleges jogkörrel rendelkező autók számára a parkoló igénybevételéhez szükséges az alkalmazásba történő regisztráció, majd helyfoglalás a rendszámukkal. Aki nem regisztrál az alkalmazásba, ő is beléphet a parkolóba, ha van szabad hely. A regisztráció és foglalás nem kötelező, de előnyt jelent. Megkülönböztetünk még egy harmadik jogkört is, ők az adminisztrátorok. Feladatuk az érkező autók beléptetése és rögzítése a rendszerben abban az esetben, ha a rendszámfelismerés és ezáltal a kapunyitás nem működne. Az alábbi használati eset diagramon ábrázoltam a jogosultságok alapján meghatározott felhasználókat és jogköreiket:



9.1. ábra: Use Case diagram a felhasználói felület funkcióihoz (Az ábra az app.diagrams.net alkalmazással készült)

9.2.Regisztráció

Az alkalmazás felhasználói számára szükséges felhasználói fiók létrehozása, ehhez a webes felületen az email címüket, teljes nevüket, telefonszámukat, és egy jelszót kötelezően meg kell adni. Lehetőségük van a regisztrációs felületen rendszámuk felvételére, abban az esetben, ha több autót használnak, akkor az összes hozzáadására a fiókjukhoz. Ez a lépés kihagyható, viszont a foglalás alkalmával kötelező lesz megadni, hogy melyik autójukkal érkeznek.

9.3.Belépés

A regisztrált felhasználók az alkalmazásba email címük és jelszavuk megadása után belépve tudnak foglalást létrehozni, a már létrehozott foglalásaikat megtekinteni, szerkeszteni, törölni, illetve újabb rendszámokat felvenni a rendszerbe.

9.4.Foglalás létrehozásának menete

A foglalás létrehozásához első lépésként ki kell választani melyik parkoló területére szeretne érkezni az adott felhasználó, megadni, hogy mely rendszámú autójával érkezik, melyik napon, milyen méretű helyet szeretne, majd ezután a rendszer automatikusan sorsol neki egy helyszámot. A parkolóban majd ezt a számú helyet kell elfoglalnia. A foglalásához eltárolásra kerül egy státusz is, ami a foglalás állapotát jelzi, ez négy értéket vehet fel:

- 0 – az alkalmazásban létre lett hozva a foglalás
- 1 – az autó megérkezett az adott parkoló bejáratához (ezt a kamerás rendszer rögzíti, vagy ha nem működne, akkor az adminisztrátor)
- 2 – az autó elhagyta a parkoló területét (szintén a kamerás rendszer vagy az adminisztrátor rögzíti)
- 3 – a foglalás nem kezdődött meg az adott napon belül, az autós nem érkezett meg

A foglalás mindig egy egész napra érvényes, ha ettől eltérően szeretne a parkolóban tartózkodni, akkor a felhasználónak kell gondoskodni az újra foglalásról, azaz, ha például egy fél nap után a kamerás rendszer érzékeli a távozását, akkor a foglalást a rendszer lezártnak tekinti, a helye újra kisorsolásra kerülhet más autósok számára, így, ha ugyanezen a napon vissza szeretne menni a parkolóba ismételten létre kell hozni egy új foglalást. Szükség esetén lehetőséget kell arra biztosítani, ha a felhasználó le szeretné mondani a foglalását akkor ezt az alkalmazáson belül meg tudja tenni a foglalás törlésével. A nap végén a meg nem érkezett autósok foglalását a rendszer lezárja, így a következő napra ezek a helyek elérhetőek lesznek.

9.5.Adminisztrátorok feladatköre

Az adminisztrátorok speciális jogkörrel rendelkező felhasználók, ők kezelik a foglalásokat, felvesznek új parkolóhelyeket a listába, szabályozzák, hogy egy parkolóban hány hely legyen elérhető foglalásra, a rendszer átmeneti meghibásodása esetén a beléptetés a feladatuk; felvinni a rendszerbe a parkolóhelyre belépő autósokat.

9.6.A parkoló területén működő rendszerrel szemben támasztott elvárások

Az autósok érkezésekor a bejáratnál, a sorompó mellett elhelyezett monitoron látható a szabad helyek száma a parkolóban, ebben a számban nem szerepelhetnek az alkalmazásban foglalt helyek és azon személyek helyei, akik rendszáma előzetesen fel van véve a rendszerben, illetve azoknak a helyeknek a darabszáma, ahol már parkol autó. Így aki foglalás nélkül érkezik, csak akkor képes bejutni a parkolóba, ha van szabad hely.

A sorompónál elhelyezett kamera egy képet készít, amin szerepel az autó rendszáma, ezt egy API-nak továbbítja, ami a képen szereplő rendszám helyét azonosítja, majd a karaktereket felismeri. Az eredményt a saját rendszerembe továbbítja, ahol az üzleti logika megállapítja, hogy a bejutni kívánó autó szerepel-e a nyilvántartásban aktív foglalással vagy előre definiált, kivételes jogkörrel rendelkező autósként, ha egyik sem teljesül, akkor rögzíti a belépés tényét. Ha a foglalás létre lett hozva, akkor ennek a rekordnak a státuszát változtatja meg.

A parkoló területére érkezve piros, illetve zöld háttérszínnel a kijelzőkön található a helyek állapota, annak megfelelően, hogy az adott hely szabad vagy foglalt – akár az alkalmazásban valaki lefoglalta, vagy fenntartott hely. A piros szín és a rendszám jelöli, hogy arra a helyre nem szabad parkolni, csak a kijelzőn szereplő rendszámú autónak, a zöld jelzi a szabad helyeket. Aki foglalt az alkalmazásban helyet, arra a jelzésű területre szükséges beállnia, amit a rendszer sorsolt neki, aki saját hellyel rendelkezik, ő értelemszerűen azt foglalja el, aki pedig foglalás nélkül érkezett, a zöld színnel jelölt helyek egyikére szükséges beállnia.

Minden helyen egy ultrahangos szenzor működik, ami érzékeli az autók jelenlétét az adott helyen, szükséges pár mérést végezni bizonyos időközönként, hogy biztosan helyes eredményt adjon, miután többszöri mérés is azt adta eredményül, hogy az adott helyen az autó jelenléte érzékelhető, akkor a beágyazott rendszer jelzi az alkalmazás számára, hogy egy autó elfoglalta ezt a helyet és rögzíteni kell adatbázisban, hogy a hely nem elérhető, ha pedig felszabadul, akkor ennek tényét.

A távozás tényét a kijáratnál elhelyezett kamera adminisztrálja ugyanolyan módon, mint a bejáratnál elhelyezett kamera, fontos, hogy a kamerákhoz hozzá kell rendelni egy azonosítót, hogy az alkalmazás külön tudja választani, hogy honnan érkezik a kép, azaz az autó éppen megérkezett vagy távozott. Tehát ismét a rendszámfelismerő API-hoz küldi a képet, ami a saját rendszerembe visszaküldi az eredményt, ahol bekerül az adatbázisba a távozás ideje és a foglalás új státusza, miután az autós távozott a parkoló területéről, a foglalást lezártnak kell tekinteni, és a helyének a száma újra kisorsolható lesz az alkalmazásban más felhasználók számára.

10. A RENDSZER FEJLESZTÉSÉHEZ VÁLASZTOTT ESZKÖZÖK ISMERTETÉSE

Az alábbi fejezetben a rendszerem megvalósításához választott szoftvereket és rendszereket mutatom be, indoklom miért ezekre esett a választásom.

10.1. Adatbázis

A rendszer megvalósításához relációs adatbázis szükséges, az adatbázisok ezen típusa jól használható strukturált adatok esetén, ami a saját rendszeremben tárolt adatokra igaz, ezek az adatok illeszkednek az előre definiált sémára. Az adatbázisban tárolni kell a felhasználók személyes adatait, a felhasználókhoz tartozó foglalásokat, a foglalások állapotával együtt, az elérhető parkolókat, ezekben a parkolóknak a helyek számát, foglaltsági állapotát, illetve a helyszínen működő rendszerben észlelt változásokat.

Az alábbi táblázatban az utóbbi időszakban használatos legnépszerűbb adatbázis kezelő rendszerek szerepelnek, ezek közül is a relációs adatbázisok relevánsak feladatom szempontjából.

Rank			DBMS	Database Model	Score		
Apr 2022	Mar 2022	Apr 2021			Apr 2022	Mar 2022	Apr 2021
1.	1.	1.	Oracle	Relational, Multi-model	1254.82	+3.50	-20.10
2.	2.	2.	MySQL	Relational, Multi-model	1204.16	+5.93	-16.53
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	938.46	+4.67	-69.51
4.	4.	4.	PostgreSQL	Relational, Multi-model	614.46	-2.47	+60.94
5.	5.	5.	MongoDB	Document, Multi-model	483.38	-2.28	+13.41
6.	6.	7.	Redis	Key-value, Multi-model	177.61	+0.85	+21.72
7.	8.	8.	Elasticsearch	Search engine, Multi-model	160.83	+0.89	+8.66
8.	7.	6.	IBM Db2	Relational, Multi-model	160.46	-1.69	+2.68
9.	9.	10.	Microsoft Access	Relational	142.78	+7.36	+26.06
10.	10.	9.	SQLite	Relational	132.80	+0.62	+7.74

10.1. ábra: Adatbázis motorok rangsora népszerűségük alapján [18]

Ezek közül az SQLite-ra esett a választásom. Ez egy C-nyelvű könyvtár, amely egy önálló, platformfüggetlen, nagy megbízhatóságú, teljes körű SQL adatbázis-motort valósít meg. A legtöbb mobiltelefonba és számítógépbe integrálva van, rengeteg alkalmazás használja. Ideális IoT projektekhez, kis-, és közepes forgalmú weboldalak adatbázisaként, mikor számít a memória és tárhely hatékony kihasználása. Az adatok helyi tárolását biztosítja az alkalmazások számára. Nem igényel semmiféle adminisztrációt, egyetlen fájlban tárolja az adatbázist. Mivel az alkalmazásban kis létszámú felhasználóra számítunk, kisebb adatmennyiséggel és kevesebb adatbázis írás művelettel, így nem okoz problémát az, hogy az SQLite egy időben csak egy írást engedélyez, ezeket a műveleteket egymás után tudja végrehajtani. Hátrányt jelenthet még, hogy nem lehetséges a felhasználók és jogosultságok kezelése, ha ezekre szükségünk van, akkor érdemes lenne egy másik megoldás választása, de a feladat ezt nem követeli meg. [19]

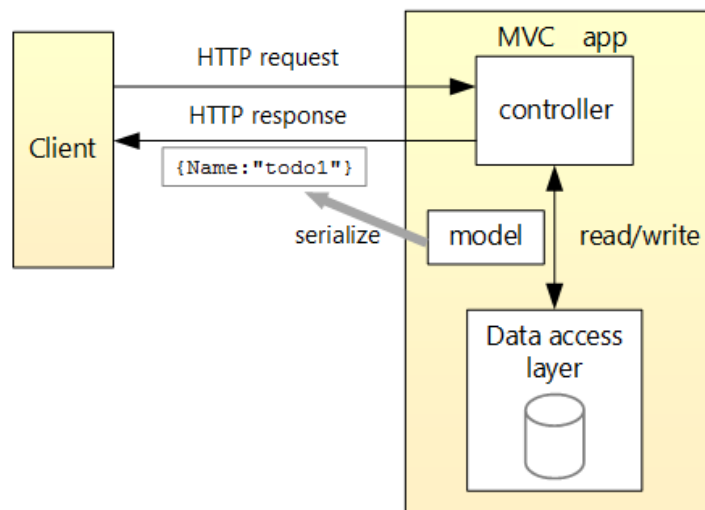
10.2. Backend

A backend feladata a frontendtől érkező kérések kezelése, a beágyazott rendszerrel való interakció, illetve az adatbázis írása-olvasása. Ennek megvalósítására a Microsoft Visual Studio fejlesztői környezetet fogom használni, ASP .NET Core Web API project template-re írom az alkalmazást, ami ASP .Net Core alkalmazások készítését teszi lehetővé controllerek létrehozásával és RESTful HTTP kérések küldésével, illetve fogadásával.

REST (Representational State Transfer) egy architektúráis stílus, célja a rendszerek kommunikációjának egyszerűbbé tétele szabványok biztosításával. Ebben az architektúráis stílusban a szerver és a kliens külön van választva, függetlenek egymástól, csak azt fontos definiálni, hogy az üzeneteket milyen formában küldjék egymásnak. Ezek a rendszerek állapot nélküliek, a szervernek nem szükséges tudnia, hogy a kliens milyen állapotban van és ez fordítva is igaz, így fel tudják dolgozni a kapott üzeneteket anélkül, hogy ismernék az eddigi kommunikációs előzményeket. A REST architektúrában a kliens kérést küld a szervernek a tárolt adatok lekérésére vagy módosítására, és a szerver ezekre a kérésekre válaszol. Egy ilyen kérés általában a következőkből áll:

- a kérés típusa (GET, POST, PUT, DELETE)
- header, a kérésről ad információt
- elérési útvonal
- body, adatot tartalmaz

A kérésekre adott válasz tartalmazhat payload-ot a kliens számára, ennek a típusát a szervernek a válasz header-jében definiálni kell. Emellett tartalmaznak státusz kódot is, amely a kliens számára ad információt a folyamat sikerességéről. [20]



10.2. ábra: Kommunikáció a kliens és a szerver között [21]

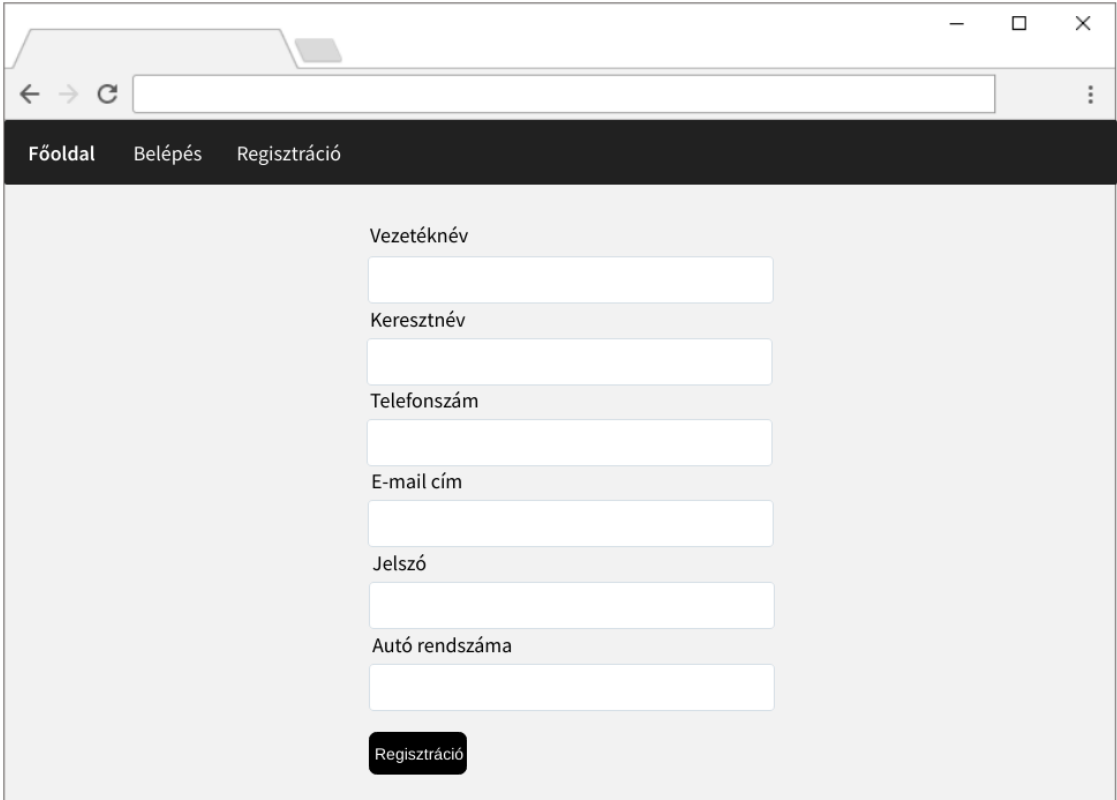
A .NET 5 keretrendszert használva elérhető a Swagger UI, ami egy webalapú interfészen teszi lehetővé az alkalmazás controllereinek tesztelését a megfelelő kérések küldésével.

Az adatok tárolásához szükséges valamilyen ORM (object-relational mapping) technika alkalmazása, ami az objektum orientált nyelven írt programkód és a relációs adatbázis típusrendszere közti konvertálásokat és megfeleltetéseket végzi. Ezt a feladatot az Entity Framework fogja ellátni.

10.3. Frontend

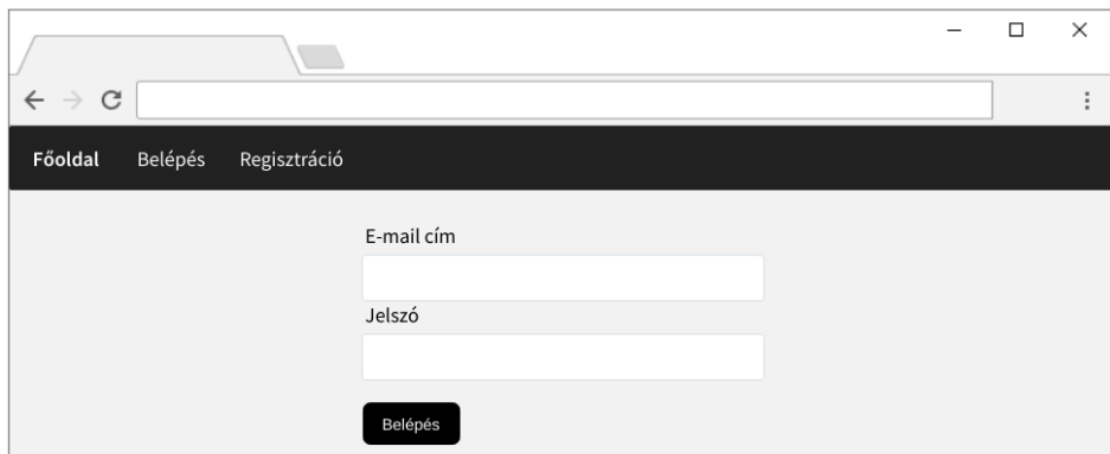
A Frontend az alkalmazás azon része, mellyel a felhasználók közvetlenül interakcióba lépnek. A fejlesztéshez az Angular keretrendszert választom. Az Angular a JavaScript ökoszisztéma része, 2009-ben a Google mutatta be és napjainkban egyre nagyobb népszerűsége tesz szert [22]. Ezen keresztül történik a regisztráció, bejelentkezés és a foglalásokkal kapcsolatos műveletek. A felhasználóbarát felület kialakításához a Bootstrap nyújt segítséget. Ezt a keretrendszert használva responsive alkalmazásokat készíthetünk, melyek többféle képernyő mérethez is igazodnak, megtartva így az eredeti esztétikus kinézetet.

Az alkalmazás tervezett kinézetét a MockFlow alkalmazással készítettem el [24]:



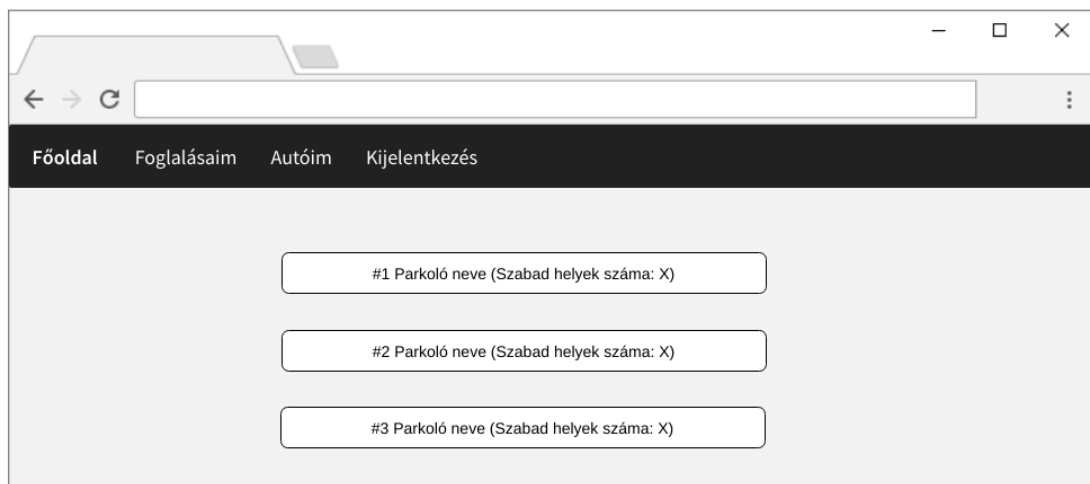
The image shows a web browser window displaying a registration form. The browser's address bar is empty. The page has a dark navigation bar at the top with three links: 'Főoldal', 'Belépés', and 'Regisztráció'. The main content area is light gray and contains a registration form with the following fields: 'Vezetéknév', 'Keresztnév', 'Telefonszám', 'E-mail cím', 'Jelszó', and 'Autó rendszáma'. Each field is represented by a white input box. Below the 'Autó rendszáma' field is a dark button with the text 'Regisztráció' in white.

10.3. ábra: Regisztrációs felület



A browser window mockup showing a login page. The browser's address bar is empty. The navigation bar is dark with three links: 'Főoldal', 'Belépés', and 'Regisztráció'. The main content area has a light gray background. It contains two input fields: the first is labeled 'E-mail cím' and the second is labeled 'Jelszó'. Below these fields is a dark button with the text 'Belépés'.

10.4. ábra: Bejelentkezési képernyő



A browser window mockup showing the main page. The browser's address bar is empty. The navigation bar is dark with four links: 'Főoldal', 'Foglalásaim', 'Autóim', and 'Kijelentkezés'. The main content area has a light gray background. It contains three rounded rectangular boxes, each with the text: '#1 Parkoló neve (Szabad helyek száma: X)', '#2 Parkoló neve (Szabad helyek száma: X)', and '#3 Parkoló neve (Szabad helyek száma: X)'.

10.5. ábra: Főoldal

A screenshot of a web browser window displaying a form for creating a reservation. The browser's address bar is empty. The navigation bar at the top contains four links: "Főoldal", "Foglalásaim", "Autóim", and "Kijelentkezés". The main content area has a light gray background. On the left side, there is a form with the following elements: a label "Parkolóház" above a text input field; a label "Autó" above another text input field; a dark gray button labeled "Új rendszám felvétele"; a label "Parkolóhely mérete" above a text input field; a label "Melyik napra foglalsz?" above a text input field; and a dark gray button labeled "Hozzáadás" at the bottom.

10.6. ábra: Foglalás létrehozása

A screenshot of a web browser window displaying a form for adding car license plates. The browser's address bar is empty. The navigation bar at the top contains four links: "Főoldal", "Foglalásaim", "Autóim", and "Kijelentkezés". The main content area has a light gray background. On the left side, there is a form with the following elements: a label "Autók" above a text input field; and a dark gray button labeled "Hozzáadás" below the input field.

10.7. ábra: Autók rendszámainak felvétele

Parkolás napja	Parkoló neve	Parkolóhely száma	Autó	Állapot	Belépés	Kilépés	Műveletek
2022.05.06.	Óbudai Campus	12	asd123	Foglalt			Módosítás Törlés
2022.05.05.	Józsefvárosi Campus	25	jkl789	Lejárt			Módosítás Törlés
2022.05.05.	Óbudai Campus	18	asd123	Befejezett	11:43:16	14:03:12	Módosítás Törlés
2022.05.04.	Óbudai Campus	56	jkl789	Befejezett	13:21:15	16:27:30	Módosítás Törlés
2022.05.04.	Józsefvárosi Campus	11	jkl789	Befejezett	08:17:24	10:22:33	Módosítás Törlés

10.8. ábra: Foglalások megtekintése

Parkolás napja	Parkoló neve	Parkolóhely száma	Autó	Állapot	Belépés	Kilépés	Műveletek
2022.05.06.	Óbudai Campus	12	asd123	Foglalt			Megérkezett Távozott Törlés
2022.05.06.	Józsefvárosi Campus	25	jkl789	Foglalt			Megérkezett Távozott Törlés
2022.05.06.	Óbudai Campus	18	qwe456	Folyamatban	11:43:16		Megérkezett Távozott Törlés
2022.05.06.	Óbudai Campus	56	tzu549	Folyamatban	09:21:15		Megérkezett Távozott Törlés
2022.05.06.	Józsefvárosi Campus	11	cvb528	Folyamatban	08:17:24		Megérkezett Távozott Törlés

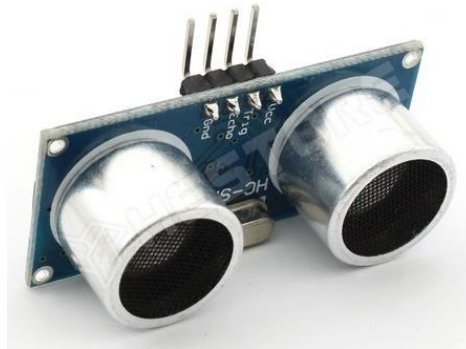
10.9. ábra: Admin felület

10.4. Beágyazott rendszer

A beágyazott rendszer feladata a helyszínre érkező autósokról információk küldése a backend réteg felé HTTP kéréseken keresztül. A hagyományos módszer szerint az érkező autósok a sorompót belépőkártyájukkal nyitják ki. A rendszerem célja, hogy a be-, és kiléptetés automatikusan történjen. Ehhez egy kamerát szükséges elhelyezni, úgy, hogy az érkező autónak a rendszámáról a sorompó előtt állva megfelelő minőségű felvételt lehessen készíteni. Azt, hogy a kamera melyik időpillanatban készítse el a képet többféleképpen meg lehet oldani, számos kamera létezik ennek a feladatnak a

megvalósítására, feladatomban a saját rendszeremet felkészíteni, hogy tudja az érkező képfájlt fogadni a kamerához rendelt azonosítóval. Az azonosító használatára azért van szükség, mert több kamerától várunk képet, a bejárat és a kijárat oldaláról, így megállapítható, hogy egy autó éppen érkezik vagy távozik. Szakdolgozatomban a kamerakép küldését a Postman alkalmazással fogom szimulálni, amely egy megfelelő szoftver webes API-k tesztelésére. A belépés és kilépés tényét tehát a rendszerben rögzíteni kell.

A parkolóban található helyek mindegyikén egy szenzor elhelyezése lenne célszerű, erre a HC-SR04-4P ultrahangos távolságérzékelőt választottam, ami 2 cm és 500 cm távolság között tud mérni. A szenzorokat a NodeMCU-ESP8266-CH mikrovezérlővel kötöm össze, ami HTTP kéréseket küld a backend-nek, ha jelenlébeli változást érzékel a megadott távolságon belül. Az alkalmazás ezeket a beérkező kéréseket kezeli, és naplózza. Minden, a foglaltság tényére vonatkozó kérés után frissíti az adott parkolóban elérhető helyek számát, amit a bejáratnál a sorompó előtt meg is jelenít egy képernyőn. A helyeken pedig szintén egy kijelzőn látható a foglaltsági állapot.

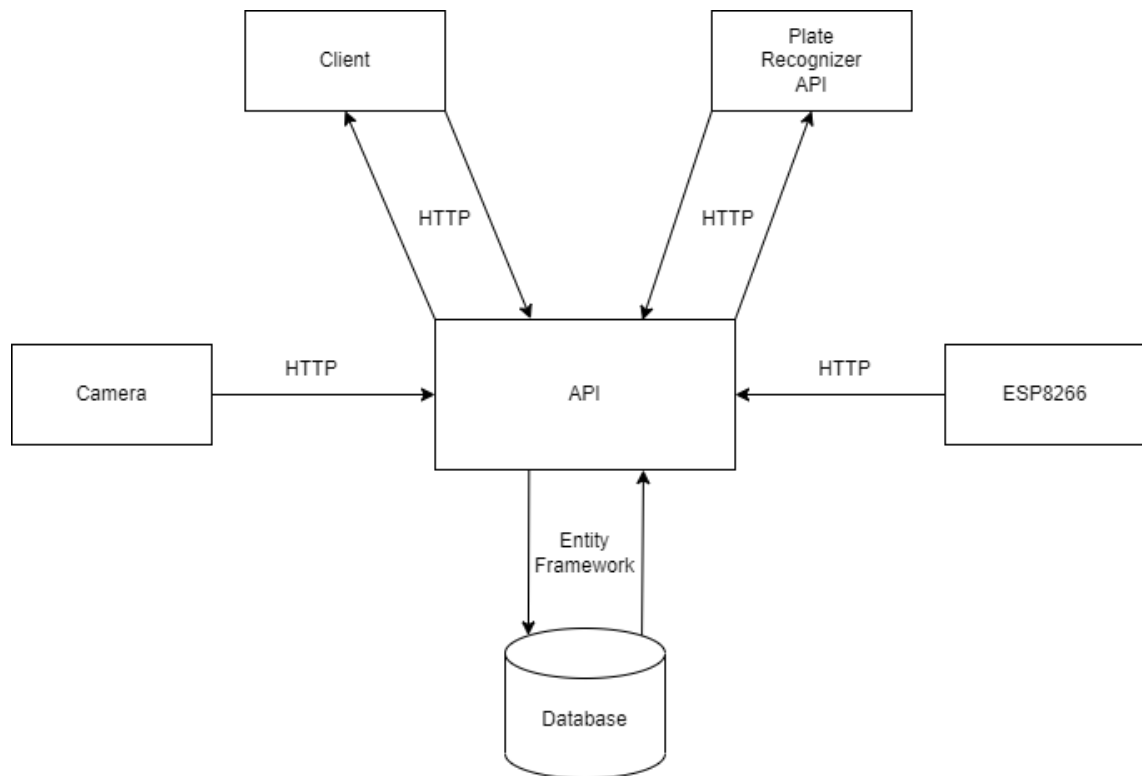


10.10. ábra: HC-SR04-4P ultrahangos távolságérzékelő modul



10.11. ábra: NodeMCU-ESP8266-CH

11.A RENDSZER FELÉPÍTÉSE



11.1. ábra: A saját rendszeremet alkotó komponensek és a közöttük lévő adatátviteli kapcsolatok (Az ábra az app.diagrams.net alkalmazással készült)

12.MEGVALÓSÍTÁS

Ebben a fejezetben a rendszerem megvalósításának lépéseit fogom bemutatni komponensenként.

12.1. Adatbázis

Az adatbázis tábláinak kialakításához meg kell határozni, hogy milyen adatokat szeretnénk tárolni:

Felhasználók

- Id: a felhasználó azonosítója
- FirstName: a felhasználó keresztnéve
- LastName: a felhasználó vezetéknéve
- UserName: a felhasználó felhasználói neve
- Email: a felhasználó e-mail címe
- PasswordHash: a felhasználó jelszava titkosított formában
- PhoneNumber: a felhasználó telefonszáma

LicensePlates

- LicensePlateText: a rendszám (ködőjel nélkül)
- UserID: a felhasználó, akihez a rendszám tartozik
- CreationDate: a létrehozás dátuma

Logs

- LogID: a bejegyzés azonosítója
- CameraID: a parkolóban elhelyezett be- vagy kijárat camera azonosítója (0-bejárat, 1-kijárat)
- FileName: a rendszámot tartalmazó képfájl neve
- IsAutomatic: jelzi, hogy az adott bejegyzés automatikusan lett létrehozva vagy adminisztrátori beavatkozással (0-adminisztrátor által, 1-automatikusan a rendszerből)
- LicensePlateText: a felismert rendszám szövege
- Region: a rendszámhoz tartozó régió
- ReservationID: a rendszámhoz tartozó foglalás
- TimeStamp: az az időpillanat, amikor az autó észlelve lett a kamerás rendszer által
- Type: az autó kategóriája

ParkingLots

- ParkingLotID: a parkoló azonosítója
- ParkingLotName: a parkoló megnevezése
- Address: a parkoló címe

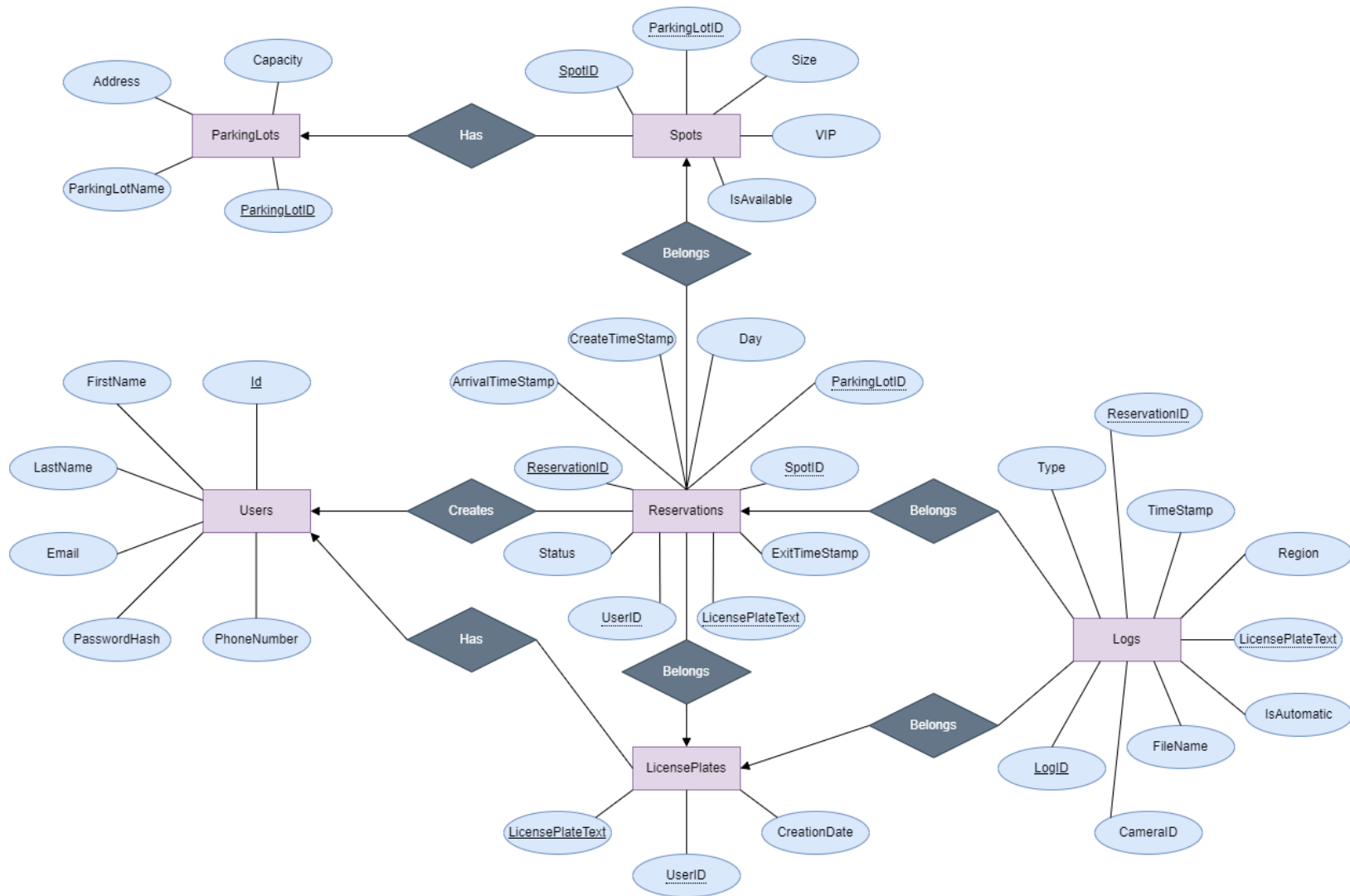
- Capacity: megmutatja, hogy hány férőhelyes a parkoló

Reservations

- ReservationID: a foglalás azonosítója
- ArrivalTimeStamp: az érkezés időpontja
- CreateTimeStamp: a foglalás létrehozásának időpontja
- Day: melyik napra szól a foglalás
- ExitTimeStamp: a távozás időpontja
- LicensePlateText: erre a rendszámra szól a foglalás
- ParkingLotID: a parkoló azonosítója
- SpotID: a parkolóban lévő hely azonosítója
- Status: létrehozott foglalás, megérkezett, távozott
- UserID: megmutatja, melyik felhasználóhoz tartozik a foglalás

Spots

- SpotID: a parkolóban található hely azonosítója
- ParkingLotID: megmutatja, hogy a hely melyik azonosítójú parkolóhoz tartozik
- Size: a hely mérete (kicsi / közepes / nagy)
- VIP: a hely különleges jogkörrel rendelkező személy számára fenn van-e tartva
- IsAvailable: a hely aktuális állapota (foglalt / szabad)



12.1. ábra: Az adatbázis egyed-kapcsolat diagramja (Az ábra az app.diagrams.net alkalmazással készült)

Az adatbázis fájl Code First módszerrel hoztam létre. A táblákat reprezentáló osztályokat és a kapcsolatukat az Entity Framework ORM eszközzel alakítottam ki.

12.2. Backend

Az alkalmazás MVVMC (Model-View-ViewModel-Controller) architektúráis mintára készült. A ViewModel-ek felelősek azért, hogy egy View fel legyen töltve adattal, illetve a View-n a felhasználtól bekért adatok is ViewModel-ben továbbítódnak az API felé a Controllereknek feldolgozásra, melyek feladata az adatbázist reprezentáló Model objektumokon a szükséges műveletek végrehajtása. Az alkalmazásban három Controller található, elkülönítve a különböző funkcionálisokat:

12.2.1. AuthController

A felhasználók hitelesítéséért és jogosultságaik kezeléséért felel.

API	Funkcionalitás	Request Body	Response Body
POST /register	új felhasználó létrehozása	RegisterViewModel	User objektum példány
POST /login	belépés felhasználói fiókba	LoginViewModel	JSON Web Token, lejárat ideje, a felhasználó admin-e
GET /get-user	felhasználó személyes adatainak lekérdezése	-	a felhasználó személyes adatai és rendszámai
POST /add-plates	rendszámok hozzáadása a felhasználóhoz	string tömb	-

12.1. táblázat: Az autentikációért felelős controller metódusai

Az alkalmazás használata regisztrációhoz van kötve, a felhasználók a felületen megadják az e-mail címüket, keresz-, és vezetéknévüket, telefonszámukat, egy jelszót, mellyel az alkalmazásba be tudnak majd lépni. Opcionálisan megadhatják a rendszámaikat is, de erre később is van lehetőségük. Regisztrációkor minden felhasználó User jogkör fog kapni. Az alkalmazás funkciói belépés után válnak elérhetővé. A felhasználók kezeléséhez a UserManager classt használtam. Mikor egy felhasználó bejelentkezik, az API egy JSON Web Token-t küld vissza, ami a felhasználó és az API közötti kommunikációt hitelesíti. A felhasználó minden HTTP kéréshez ezt a Token-t hozzáfüzi, és a rendszer ez alapján engedélyezi vagy tiltja a hozzáférést bizonyos API metódusokhoz [23]. A másik felhasználói jogkör az Admin jogkör, ő a bejelentkezés során Adminként lesz hitelesítve, mert előzetesen fel lettek véve az adatbázisba.

Ez a controller felel még a felhasználók adatainak szolgáltatásáért, a kliens felé ViewModel formájában küldi az éppen bejelentkezett felhasználó adatait, így a felhasználók meg tudják tekinteni személyes adataikat, rendszámaikat. Lehetőségük van rendszámok hozzáadására a felhasználójukhoz, ha több autóval rendelkeznek és várhatóan ezeket váltogatva foglalnak helyet.

12.2.2. ReservationController

A foglalásokkal kapcsolatos HTTP kérések ide érkeznek be.

API	Funkcionalitás	Request Body	Response Body
GET /	visszaadja az adatbázisban szereplő összes foglalást	-	Foglalásokat tartalmazó lista
GET /user	visszaadja egy felhasználóhoz tartozó foglalásokat	felhasználó azonosítója	Foglalásokat tartalmazó lista
GET /reservation/{id}	egy foglalás adatait adja vissza	a foglalás azonosítója	Foglalás objektum
GET /parkingLot/{id}	egy parkolóra vonatkozó foglalásokat adja vissza	a parkoló azonosítója	Foglalásokat tartalmazó lista az adott parkolóra
POST /alpr	a beérkező képet elküldi a Plate Recognizer API-nak, majd a visszakapott választ feldolgozza	képet, kamera azonosítóját és parkolót tartalmazó ViewModel	A bejáratnál megjelenítendő üzenet
POST /	foglalás létrehozása	parkoló azonosítóját, parkolóhely méret igényét, napot és rendszámot tartalmazó ViewModel	a foglalt hely száma és mérete
PUT /	foglalás szerkesztése	foglalás azonosítóját, rendszámot és napot tartalmazó ViewModel	az új hely száma

DELETE /{id}	foglalás törlése	a foglalás azonosítója	-
PUT /editStatus	foglalás státuszának beállítása	a foglalás azonosítóját és státuszát tartalmazó ViewModel	-

12.2. táblázat: A foglalások kezeléséért felelős controller metódusai

A felhasználó a saját fiókjába belépve tud foglalást létrehozni valamelyik rendszámára egy általa választott parkolóban. Ki tudja választani, hogy mekkora méretű helyre van szüksége: kicsi, közepes vagy nagy. A helyet mindig egy adott napra lehet foglalni. Ha nem lenne a megadott paraméterekkel szabad hely a parkolóban, akkor erről azonnal értesítést kap, nem jön létre a foglalás. Meg tudja tekinteni foglalásait, tudja ezek paramétereit módosítani: a rendszámot, és azt, hogy melyik napra tart igényt a helyre, ha a hely méretét módosítani kívánja, akkor törölnie kell a foglalást és újat létrehozni, hiszen amit a sikeres foglaláskor kapott helyszámot, már nem fog egyezni a megváltoztatott mérettel. A felhasználó, ha egy napra már foglalt helyet, akkor ugyanezzel a rendszámmal már nem tud megint foglalást létrehozni erre a napra, ha a foglalás státusza nem befejezett. Tehát ha egyik napra foglalt, megérkezett a parkolóba, majd elhagyta a parkoló területét, akkor a foglalását lezártnak tekintjük, és ha szeretne újra visszamenni a parkolóba ugyanezen a napon később foglalással, akkor megint igényelnie kell a helyet. Mindig csak akkor lehet módosítani vagy törölni a foglalást, ha ennek státusza még meg nem kezdett.

Ez a controller kezeli a foglalások állapotváltozásait is. Ez három állapotot vehet fel:

- 0 – le van foglalva az alkalmazásban
- 1 – a parkolást elkezdte
- 2 – befejezte a parkolást
- 3 – érvénytelen, az adott rendszámú autó nem érkezett meg a parkolóba

Ha az 1-es vagy 2-es státusz nem a kamerás rendszer által lett visszaadva, akkor adminisztrátori beavatkozás történt, ilyenkor a Log táblában a kamera azonosítója a be-, és kijárat kameráktól eltérő azonosítót kap, jelezve ezzel, hogy nem ismerte fel automatikusan a rendszer a rendszámot. Fontos információként bekerül még a rendszám és az idő, amikor a belépés történt. A foglalások táblájába pedig a foglalás státusza is beállítódik.

12.2.3. ParkingLotController

Ennek a controllernek a feladata a parkolóban lévő helyek menedzselése. A parkolóhelyekről lehet tudni, hogy azok melyik parkolóban találhatók.

API	Funkcionalitás	Request Body	Response Body
GET /message/{parkingLotId}	A megadott parkolóhoz tartozó üzenetet adja vissza	parkoló azonosítója	a parkolóhoz tartozó üzenet
GET /	Visszaadja az összes parkolót	-	A parkoló adatait tartalmazó ViewModel
POST /	új parkoló felvétele	parkoló nevét, kapacitását, címét tartalmazó ViewModel	ParkingLot példány
PUT /	parkoló adatainak szerkesztése	parkoló nevét, helyek számát, címét tartalmazó ViewModel	-
GET /AvailableSpots/{parkingLotId}	elérhető helyek számának lekérése adott parkolóban	parkoló azonosítója	a parkolóban lévő szabad helyek száma
GET /Spot/{spotId}	visszaadja egy hely foglaltsági állapotát	a hely azonosítója	hely azonosítóját, elérhetőségi állapotát, szöveges üzenetet és napot tartalmazó ViewModel
POST /change-status	parkolóhely foglaltsági állapotának megváltoztatása	a hely azonosítóját és a beállítandó állapotot tartalmazó ViewModel	-

12.3. táblázat: A parkoló adatainak kezeléséért felelős controller metódusai

A felhasználók a felületen le tudják kérni az adott parkolóházakra vonatkozó információkat: nevük, címük és hogy hány szabad hely van még. Az adminisztrátori jogkörrel rendelkező felhasználók tudják bővíteni az elérhető parkolók listáját. Ha új parkolót akarnak hozzáadni a rendszerhez, akkor szükséges megadniuk, hogy az új parkolóban milyen arányban vannak a kicsi és nagy helyek, a közepes méretű helyek számát nem szükséges megadni, mert minden hely alapértelmezetten közepes. Ezen kívül a különleges jogkörrel rendelkező autósok számára vannak fenntartott helyek a parkolóban, ezek konkrét kijelölt helyek és a rendszámokhoz tartoznak, az egyszerű felhasználók nem foglalhatják el, ezek a helyek nem kerülnek kiosztásra sem az alkalmazás által, az ilyen helyeken a kijelzőn mindig a „Fenntartott hely” szöveg fog megjelenni. Ez a controller kezeli még a beágyazott rendszertől érkező HTTP kéréseket is, a szenzorok, ha állapotváltozást érzékelnek, küldenek egy kérést az API-nak, ami tartalmazza a parkolóhely azonosítóját és hogy az adott hely új állapota foglalt vagy szabad. Az API ennek megfelelően újra számolja az elérhető helyeket és növeli vagy csökkenti a foglalható helyek számát a felhasználók részére.

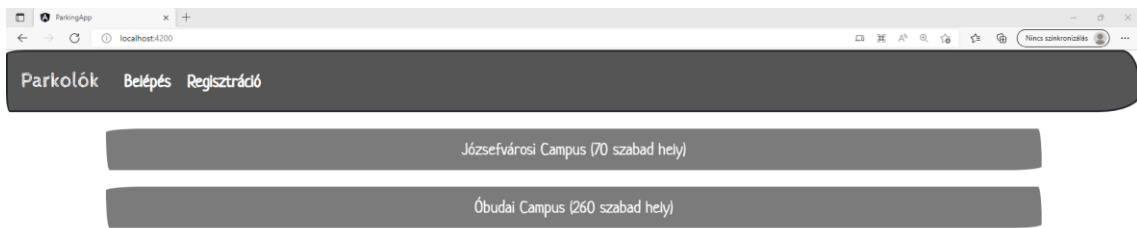
12.2.4. Services

A parkolókhöz és a helyekhez tartozó üzenetek írására és olvasására szolgáltatásokat (Service) hoztam létre, így az itt található metódusokhoz több controller is hozzá tud férni. Az egyik szolgáltatás a parkolók üzeneteit írja és olvassa, a másik pedig a parkolóhelyekhez tartozó üzeneteket menedzseli. Ezt a két szolgáltatást a ReservationController és a ParkingLotController veszi igénybe, céljuk a felhasználók számára érthető tájékoztatás nyújtása az alkalmazás használata közben, illetve a helyszínen elhelyezett tájékoztató kijelzőkön.

A ParkingLotService-nek két metódusa van a parkolókhöz tartozó üzenetek beállítására és olvasására. A ParkingSpotService-ben található az a metódus, amely eldönti egy megadott helyről, hogy VIP, tartozik-e hozzá létrehozott vagy megkezdett foglalás, áll-e autó a helyen, vagy a hely szabad, ezek alapján beállít a helyekhez egy szöveges üzenetet, melyet a helyeken elhelyezett kijelző fog mutatni. Ebben a service-ben lehet lekérni még egy adott parkolóhelyre a szabad helyek számát, amit úgy számol ki, hogy a nem elérhető helyeket (VIP, elfoglalt, vagy az alkalmazásban lefoglalt) kivonja a parkolóban lévő összes helyből. Ez a service felel a helyek kiosztásáért is a felhasználó által megadott paraméterek alapján. Itt történik még a lefoglalt, de az adott napon meg nem érkezett felhasználók foglalásának törlése a következő napon.

12.3. Frontend

A Frontend fejlesztéséhez az Angular keretrendszerben dolgoztam. Az alkalmazás webes felületen elérhető a felhasználók számára. Az alkalmazás megnyitásakor a főoldalon a parkolók listája található, mellettük megjelenik az aktuálisan szabad helyek száma is, ezt az oldalt a nem regisztrált felhasználók is meg tudják tekinteni, azonban, ha foglalást szeretnének beküldeni, regisztrálniuk, majd belépniük kell.



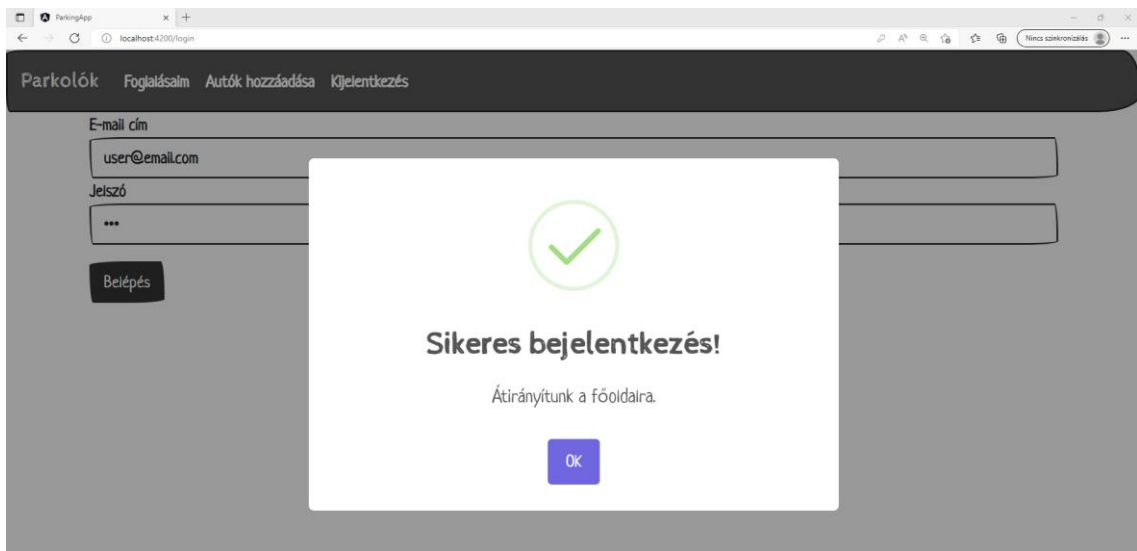
12.2. ábra: A főoldal belépés előtt

A regisztrációs képernyőn a következő adatokat kell megadni:

12.3. ábra: Regisztrációs felület

Ha már regisztráltunk az alkalmazásba, be tudunk lépni:

12.4. ábra: Bejelentkezési képernyő



12.5. ábra: Sikeres bejelentkezés

Ha a belépés sikeres volt, megjelenik a főoldal, most már a parkolók nevei aktívak, ki lehet választani. Ha a kurzort hosszan a parkoló neve felett tartjuk, megjelenik a parkoló címe.



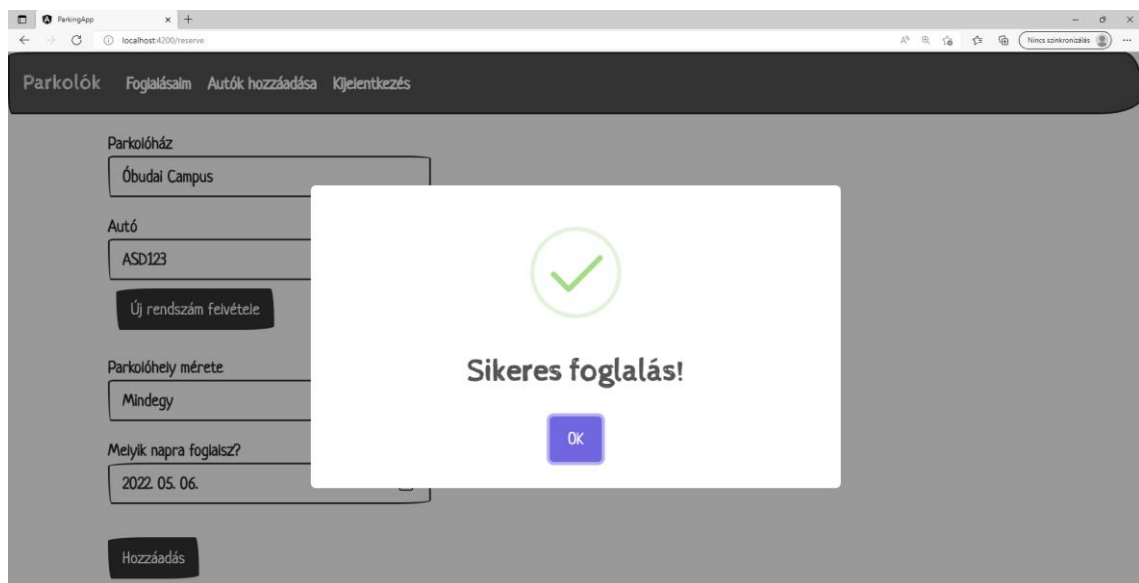
12.6. ábra: Főoldal bejelentkezett felhasználók számára

A következő felületen tudunk helyet foglalni, ehhez meg kell adni a hely méretének igényét (Mindegy / Kicsi / Normál / Nagy), az autó rendszámát, amelyikkel érkezni szeretnénk. illetve, hogy melyik napra szóljon a foglalás.

The screenshot shows a web browser window with the URL `localhost:4200/reserve`. The application has a dark header with navigation links: **Parkolók**, **Foglalásaim**, **Autók hozzáadása**, and **Kijelentkezés**. The main form contains the following fields and buttons:

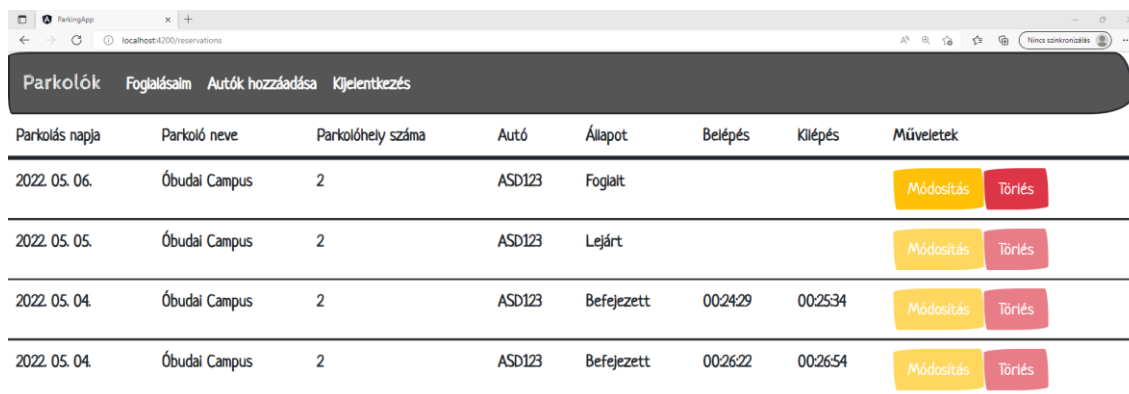
- Parkolóház:** A text input field containing "Óbudai Campus".
- Autó:** A text input field containing "ASD123".
- Új rendszám felvétele:** A dark button.
- Parkolóhely mérete:** A text input field containing "Mindegy".
- Melyik napra foglalsz?:** A date picker showing "2022. 05. 06." with a calendar icon.
- Hozzáadás:** A dark button at the bottom.

12.7. ábra: Foglalás létrehozása



12.8. ábra: Visszajelzés a foglalás sikerességéről

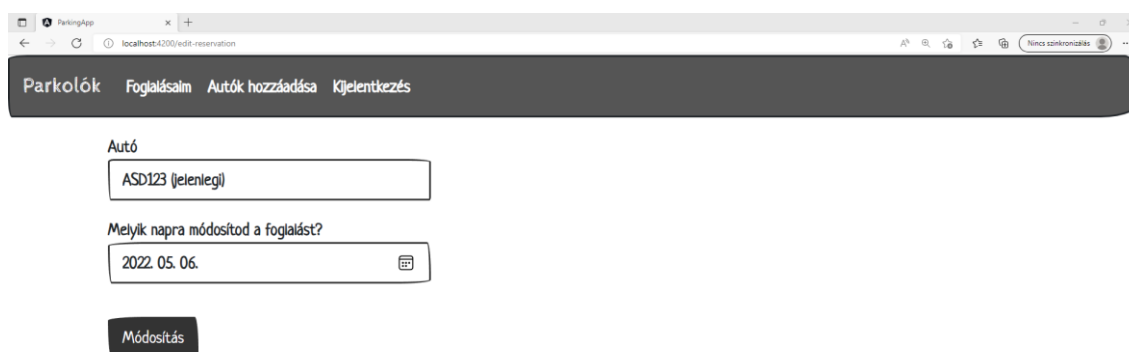
Amint leadtuk a foglalást, az alkalmazás átirányít a saját foglalásaimra, ahol szerepel az összes eddig leadott foglalás, itt meg lehet tekinteni a foglalások információit, szerkeszteni vagy törölni a még meg nem kezdetteket. Azoknál a foglalásoknál, amiket nem lehet szerkeszteni a gombok nem is elérhetőek.



Parkolás napja	Parkoló neve	Parkolóhely száma	Autó	Állapot	Belépés	Kilépés	Műveletek
2022. 05. 06.	Óbudai Campus	2	ASD123	Foglalt			Módosítás Törés
2022. 05. 05.	Óbudai Campus	2	ASD123	Lejárt			Módosítás Törés
2022. 05. 04.	Óbudai Campus	2	ASD123	Befejezett	00:24:29	00:25:34	Módosítás Törés
2022. 05. 04.	Óbudai Campus	2	ASD123	Befejezett	00:26:22	00:26:54	Módosítás Törés

12.9. ábra: A bejelentkezett felhasználó foglalásai

Ezen az oldalon lehet módosítani egy foglalást. Fontos, hogy csak a rendszámot és a napot lehet szerkeszteni, ha már a hely méretét is szeretnénk szerkeszteni, akkor új foglalást kell létrehozni, mert új helyszám kiosztására van szükség, hogy a mérete más legyen.



Autó

ASD123 (jelenlegi)

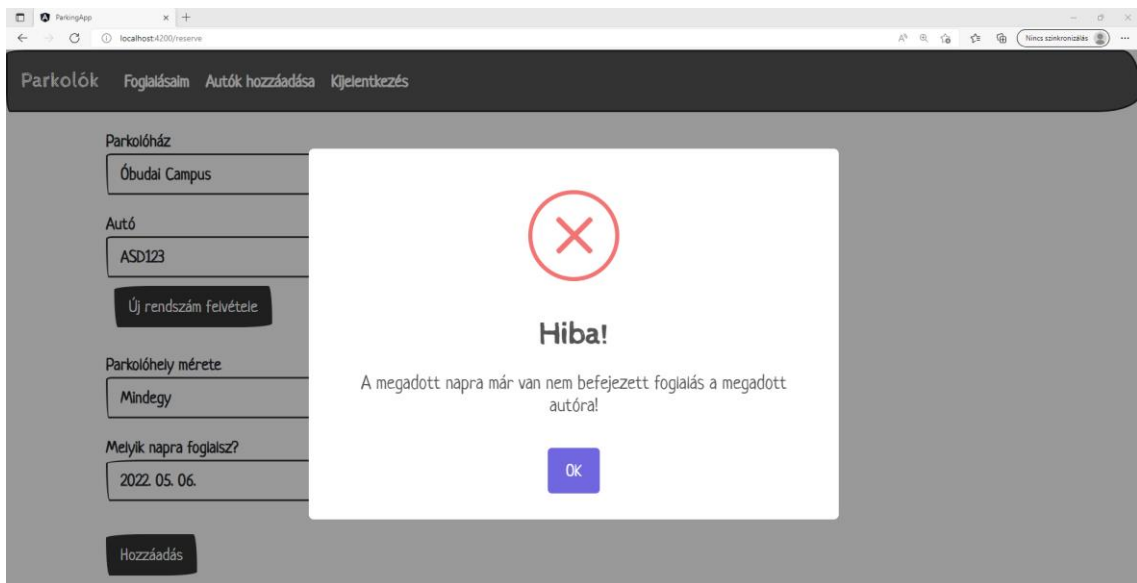
Melyik napra módosítod a foglalást?

2022. 05. 06.

Módosítás

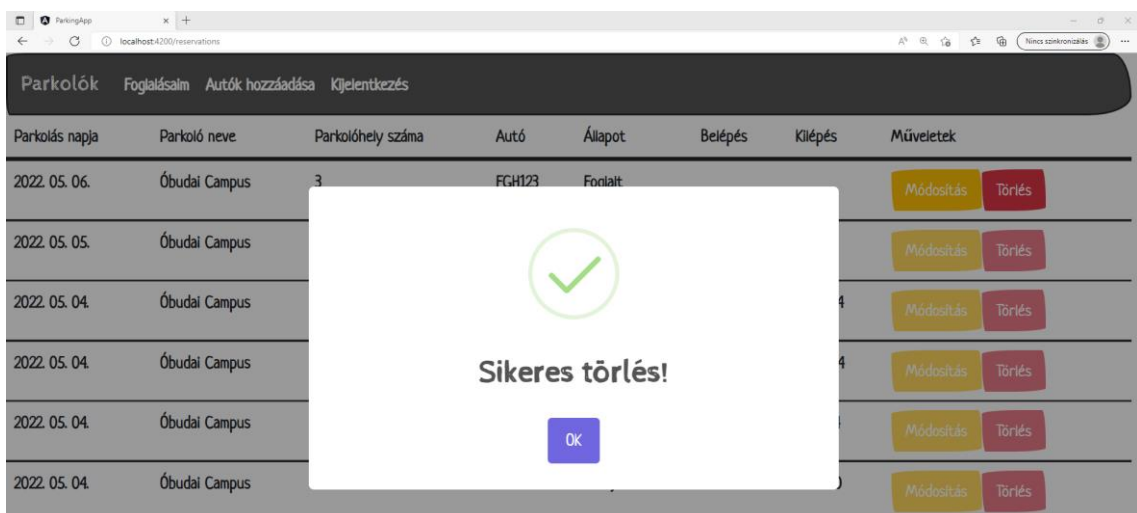
12.10. ábra: Foglalás módosítása

Ha most megint ugyanebbe a parkolóba ugyanerre a napra, ugyanezzel a rendszámmal szeretnénk helyet foglalni, akkor az nem lehetséges, a következő hibaüzenetet kapjuk:



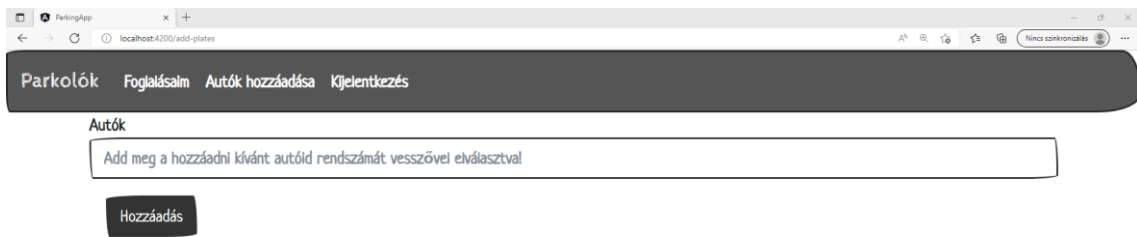
12.11. ábra: A megjelenő üzenet, ha nem sikerül a foglalás

Az új foglaláshoz először törölni kell a már meglévőt, azonban ez csak akkor lehetséges, ha státusza még nem megkezdett foglalás.



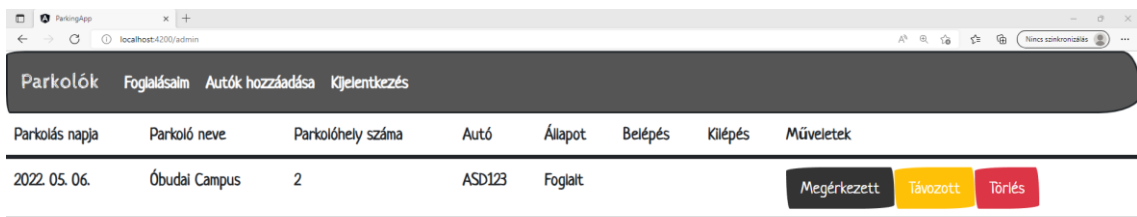
12.12. ábra: Üzenet a foglalás törlésének sikerességéről

Van lehetőség rendszámok felvételére a felületen, ezt három helyről is el lehet érni: regisztrációkor, foglalás beküldésénél az „új rendszám felvétele” gombra kattintva, és a menüsorban az „Autók hozzáadása” menüpontra kattintva.



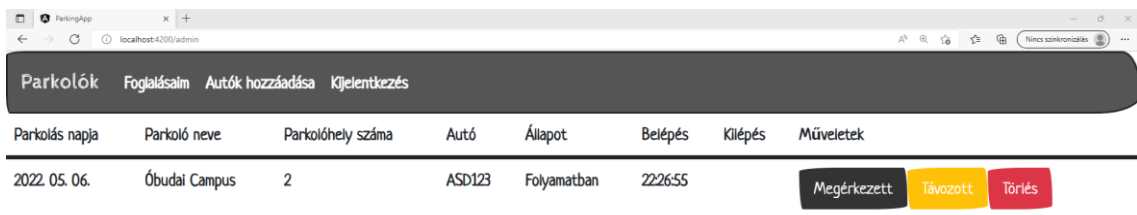
12.13. ábra: Rendszám felvétele

Az admin felületen pedig a beérkezett foglalások státuszát tudjuk beállítani, erre a kamerás rendszer átmeneti meghibásodása esetén lehet szükség, illetve, ha nem ismerné fel a rendszámot, és már nincs szabad hely, de a felhasználó foglalással rendelkezik.



12.14. ábra: Admin felületen megjelenik a foglalt státuszú foglalás

A foglalást megérkezettre állítva bekerül a belépés ideje



12.15. ábra: Az admin a „Megérkezett” gombbal „Folyamatban” státuszúra tud állítani egy foglalást

Távozttra állítva pedig a kilépés dátuma kerül be, és az admin listájából törlődik is a foglalás, hogy mindig csak azok jelenjenek meg, melyeken szükséges lehet módosításokat végrehajtani.

12.4. A parkoló területén működő rendszer

12.4.1. Kamerás rendszer

A kamera szimulálása a Postman alkalmazáson keresztül történik, ez küld egy Post kérést az API-nak, amely egy ViewModelt fogad, ez a következőket tartalmazza:

- egy IFormFile interface-t megvalósító objektum formájában a képfájlt,
- az adott kamerához tartozó azonosítót (be- vagy kijárat camera),

- a parkolóház azonosítóját.

A controller metódusa ezt a „képet” base64 string byte tömbbé konvertálja, erre azért van szükség mert a Plate Recognizer API-ja ilyen formátumban tudja feldolgozni. Ezt, és a kamera azonosítóját a megfelelő url-re, a felhasználóhoz tartozó tokenel POST kérés formájában elküldi. A választ JSON objektumként kapom vissza, amit a feldolgozáshoz deserializálni szükséges.

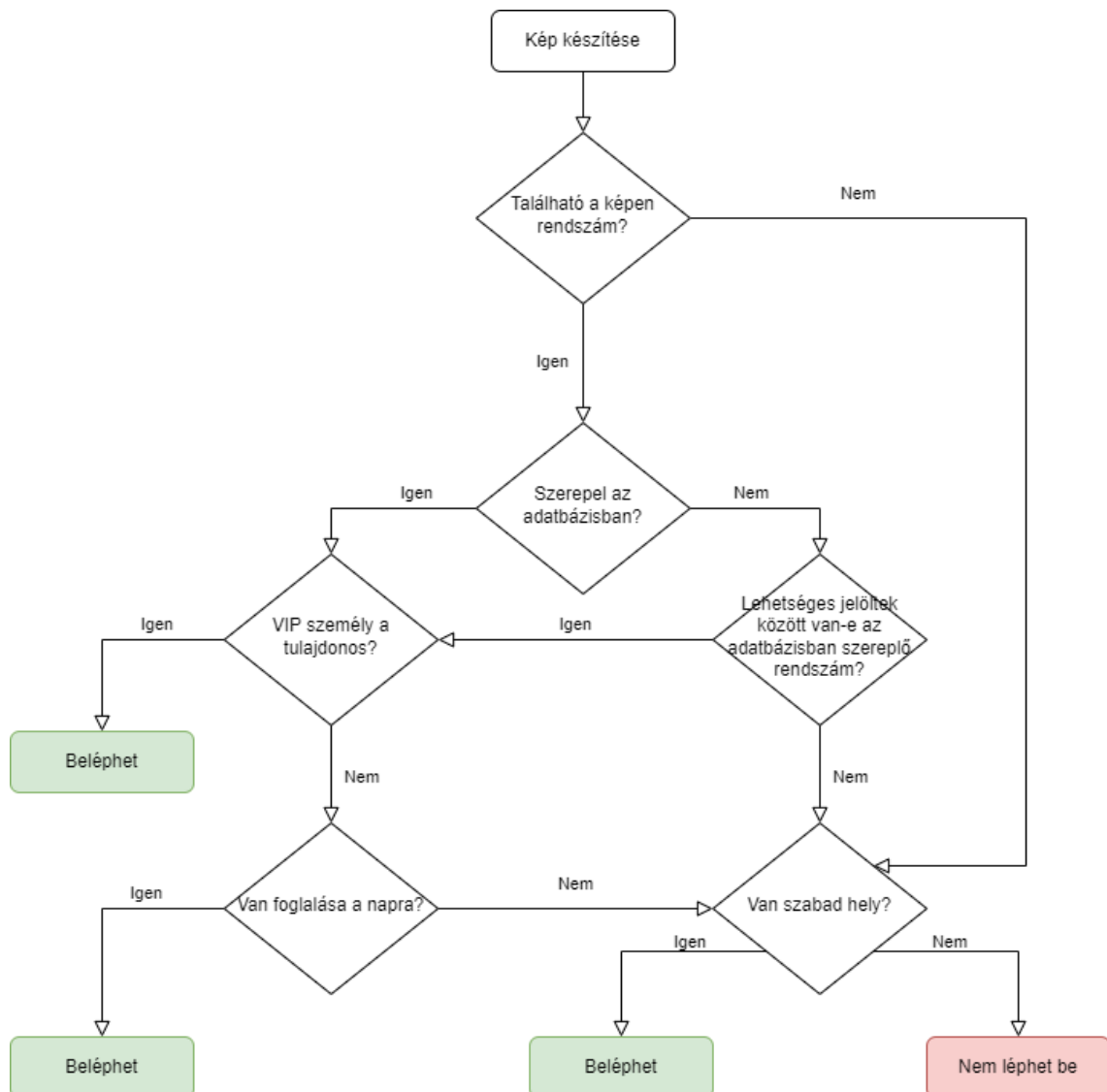
Az érkező autó rendszámára az alábbiak közül valamelyik eset lehet érvényes:

- I. Az autó tulajdonosa rendelkezik az adott napra a parkolóba aktív foglalással
A Plate Recognizer által visszaadott értékeket elmentjük az adatbázis Log táblájába. Ezután szükséges megvizsgálni, hogy melyik kamera küldte a képet. Ha a bejáratnál készült a kép, akkor a foglalás státuszát megkezdettre, az érkezés idejét a kép készülésének idejére állítjuk. Ezen kívül még a lefoglalt hely elérhetősége is foglaltra változik. Ha a kijáratnál érzékelte az autót, akkor a foglalás státusza lezárt lesz, a kilépés ideje bekerül a foglaláshoz. A helye pedig felszabadul.
- II. Az rendszám egy különleges jogkörű személyhez tartozik
Nekik nem szükséges helyet foglalni, fenntartott hellyel rendelkeznek, ami azt jelenti, hogy ezek a helyek az alkalmazásban nem kerülhetnek kiosztásra, a helyszínen táblán szükséges feltüntetni, hogy kihez tartozik a hely, hogy mások ne álljanak oda. Az ő adataik is bekerülnek a Log táblába.
- III. A rendszámhoz nem található aktív foglalás
Amennyiben a parkolóban van szabad hely, az illető beléphet. Az adatai rögzítésre kerülnek az adatbázisban.
- IV. Nem található rendszám a képen
Ekkor nem fog felismerni rendszámot, ezáltal nem is tud foglalást lekérni. Ha van szabad hely, beléphet a parkoló területére, bekerül az adatbázisba a belépés ideje és a kamera azonosítója.

Kilépésnél rendszámtól függetlenül mindenkinek kinyílik a sorompó. Belépésnél pedig csak akkor, ha van szabad hely. Ha nincs, akkor csak a foglalt vagy fenntartott hellyel rendelkező személyek léphetnek be.

Elmondható, hogy a felismert rendszám, az aktuális nap és az aktuális parkoló alapján az alkalmazás az adatbázisból lekéri a foglalást. Ha nem tartozik hozzá foglalás, akkor az történhet amiatt, hogy a Plate Recognizer nem ismerte fel helyesen a képen szereplő rendszámot. Ebben az esetben a válaszként visszaadott candidates lista tartalmazza azokat a rendszámokat, amelyek valamilyen valószínűséggel egyezhetnek az adatbázisban szereplő foglalásokhoz tartozó rendszámokkal. Ezen a listán végig kell menni, amíg meg nem találja valamelyik rendszámot aktív foglalással az adott napra az adott parkolóba. Ha megtalálja, akkor a fenti esetek közül az I. érvényesül. Ha nem találja, akkor ez egy olyan

eset, mikor az alkalmazásban foglalás nem történt, ekkor az eredetileg felismert rendszám fog bekerülni az adatbázisba. Szükséges még a kamera azonosítója alapján beállítani az érkezés vagy távozás idejét, és a foglalás státuszát.

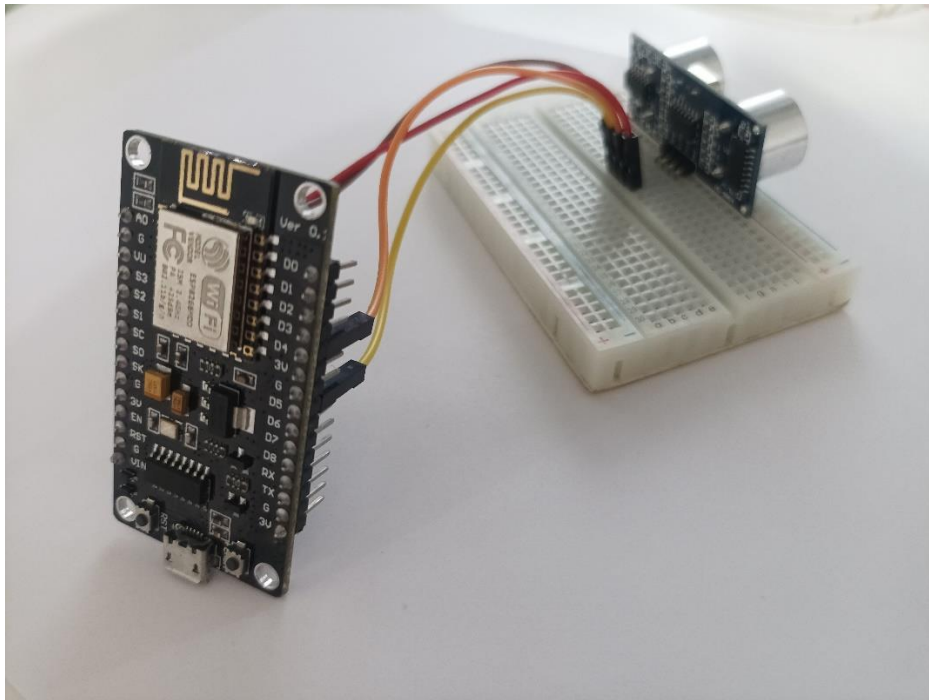


12.16. ábra: A Plate Recognizer által visszaadott eredmény alapján a beléptetés folyamatábrája (Az ábra az app.diagrams.net alkalmazással készült)

12.4.2. Parkolóhelyek foglaltsági állapotát jelző rendszer

A parkolóhelyek aktuális állapotáról a mikrovezérlőhöz csatlakoztatott ultrahangos szenzorok adnak információt. Az ESP8266 mikrovezérlőhöz a tesztelés céljából egy HC-SR04 ultrahangos szenzort csatlakoztattam. Ez másodpercenként megméri, hogy a legközelebbi objektum tőle hány cm-re található, ha három egymást követő mérés eredménye azonos intervallumba esik, és a helyen státusz változás következik be, akkor

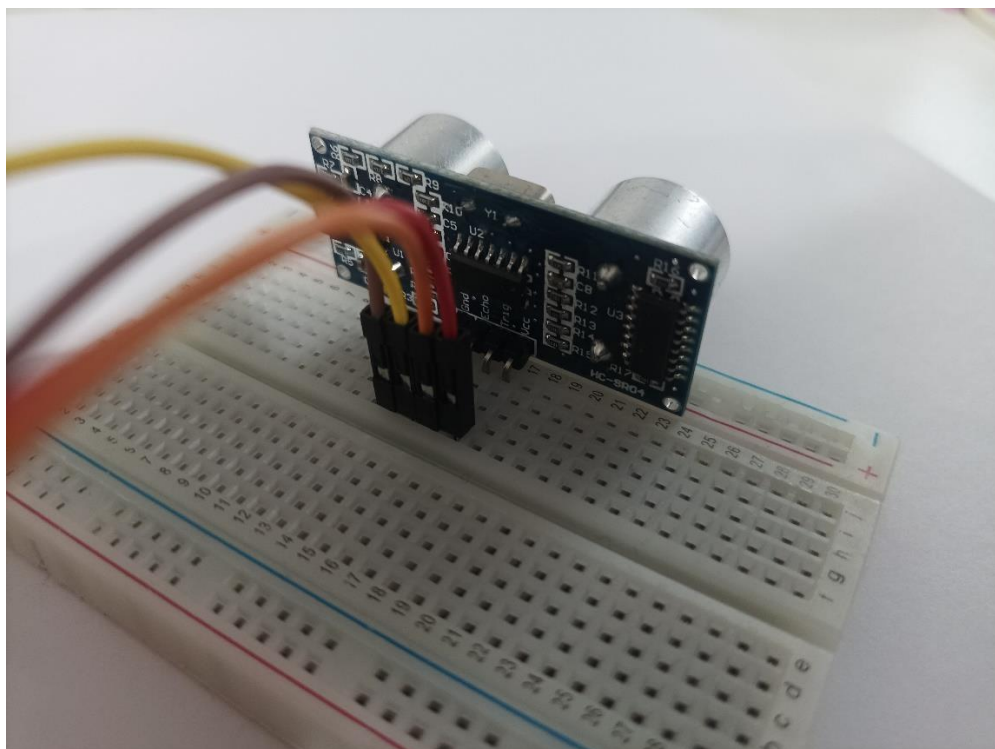
küld egy POST kérést az API-nak az aktuális foglaltsági állapottal és a hely számával. A távolság mérésénél 50 cm alatti érték esetén foglaltnak tekintjük a helyet, ennél nagyobb távolság esetében pedig szabadnak.



12.17. ábra: Az echo és trig pin-ek a D5 és D4 pin-ekhez csatlakoznak



12.18. ábra: A szenzort 5 V feszültséghez és a GND-hez van csatlakoztatva



12.19. ábra: Az ultrahangos szenzorhoz csatlakozó kábelek

12.4.3. A parkoló területén lévő tájékoztató rendszer

A parkoló előtt egy kijelzőn szükséges megjeleníteni a szabad helyek számát, így aki foglalás nélkül érkezne már a bejáratnál tájékozódik a parkoló telítettségéről. A megjelenő üzenetek szinkronizációját a SignalR könyvtár funkcióival oldottam meg, hogy a backend üzeneteket tudjon küldeni a kliensek, amikor a megjelenítendő információk frissítésére van szükség.

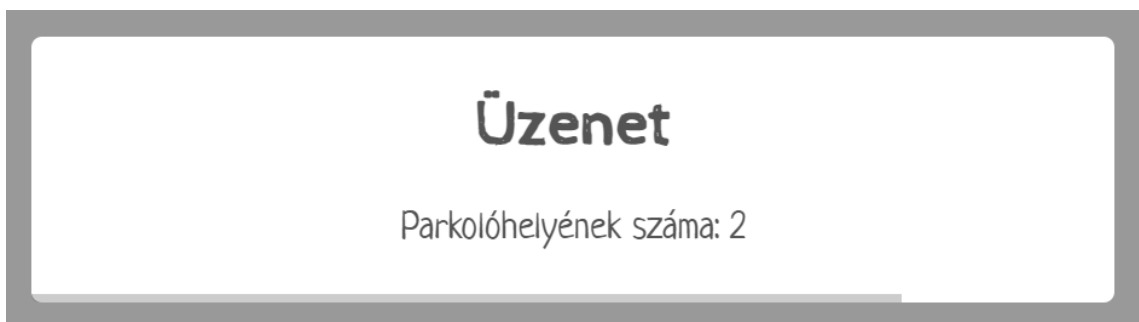


12.20. ábra: A parkolóban vannak szabad helyek

0 SZABAD HELY

12.21. ábra: A parkolóban nincs már szabad hely

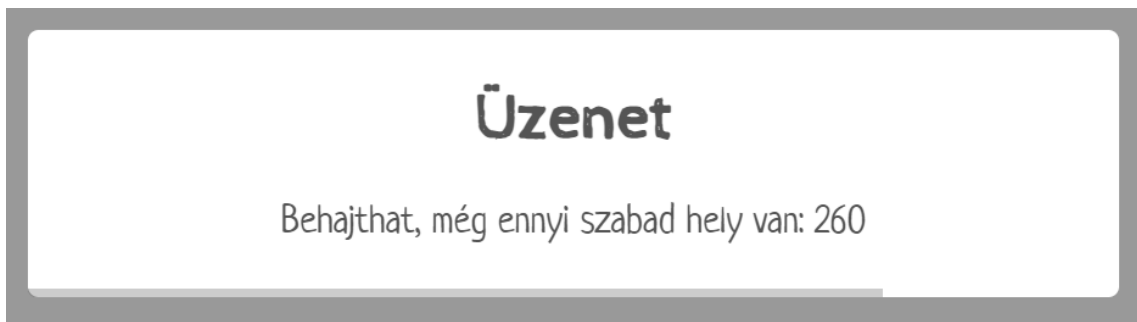
Belépésnél a kapu előtt a rendszámfelismerő rendszer az érkező autóról képet készít, majd miután feldolgozta a képet a következő üzenetek valamelyikét fogja megjeleníteni pár másodpercig:



12.22. ábra: Az adott rendszámmal foglaltak helyet az alkalmazásban, ennek a számát jeleníti meg



12.23. ábra: A fenntartott hellyel rendelkező személy foglalás nélkül beléphet



12.24. ábra: A rendszámmal nem foglaltak helyet, de még van szabad hely a parkolóban, ezért beléphet



12.25. ábra: Az érkező autósnek nincs foglalása, nem rendelkezik fenntartott hellyel és a parkolóban nincs már szabad hely, így nem léphet be

A parkolóban minden helynél szükséges egy kijelzőn megjeleníteni az adott hely állapotát, amely lehet:



12.26. ábra: A hely tulajdonosa különleges jogkörrel rendelkező személy



FOGLALT

12.27. ábra: A kijelzőn "Foglalt" szöveg jelenik meg, ha az autós foglalás nélkül érkezett és beállt az egyik helyre



ASD123

12.28. ábra: Az alkalmazásban foglalást beküldő autós helye



SZABAD HELY

12.29. ábra: A hely szabad

13.TESZTELÉS

A tesztelés során a rendszer működését vizsgálom, hogy hogyan reagál bizonyos bemeneti paraméterekre. A megadott teszteseteknek két kimenete lehet: a teszt sikeres, az elvárt működésnek megfelelő, vagy a megadott bemeneti paraméterre nem az elvárt módon reagál, ekkor a teszteset nem sikeres. A teszteléshez egy gyakori mintát, az AAA felépítést fogom alkalmazni. Ennek három lépése van:

- Arrange: első lépésként az alkalmazást kell előkészíteni, objektumokat létrehozni, beállításokat elvégezni, amennyiben szükséges
- Act: a második lépésben a tesztelendő egyetlen lépést végrehajtjuk, ez lehet például egy metódus hívása, REST API kérés küldése
- Assert: az Act lépésben kapott eredmény vizsgálása, az elvárt működésnek megfelelő vagy nem, annak eldöntése, hogy az adott teszt sikeres vagy nem

A legelterjedtebb programozási nyelvekhez mindig van legalább egy teszteléshez alkalmas keretrendszer. Az elkészített rendszerem API teszteléséhez az xUnit.net keretrendszert használom, ez egy nyílt forráskódú, unit teszteléshez alkalmas .NET keretrendszer. Az unit tesztekkel a szoftverek legkisebb logikai egységei tesztelhetők, mint a metódusok, függvények. [24] [25]

A controller függőségeit Moq keretrendszer segítségével helyettesítettem, erre azért volt szükség, hogy a tesztelés során magát a kódot teszteljem és ne a controller függőségeit.

Az adatbázist az Entity Framework Core In-Memory Database Provider-rel helyettesítettem, hogy ne az éles adatbázison futtassam a teszteket, mert az API sok esetben összetettebb lekérdezéseket használt, aminek a kezelésére a Moq önmagában nem volt ideális.

A felhasználói felület teszteléséhez End-to-end teszteket fogok végezni, ez a tesztelési típus a felhasználói viselkedésre épül, azt vizsgálja, hogy a különböző felhasználási esetekre hogyan reagál a rendszer. [26]

13.1. API tesztek

Az API tesztelés során teszteltem a foglalások, illetve a parkolók kezeléséért felelős Controllereket, valamint a hozzájuk tartozó Service-t. Ez magába foglalta az írás, olvasás műveleteket, azonban csak a Controller felelősségi körébe tartozó funkciókat vizsgáltam, azaz a beszúrás és módosítás műveleteknél magának a műveletnek a meghívását vizsgáltam, ennek az adatelérési rétegben történt eredményét nem. A 13.1.-es ábrán látható, hogy a létrehozott tesztek sikeresek voltak. Az xUnit-nak köszönhetően egy teszthez több tesztesetet meg lehetett adni, ez pedig támogatta a minél több lehetőséget lefedő tesztesetek írását.

▲  ParkingLotControllerTests (7)	245 ms
 AddParkingLot_Inserts_Record	7 ms
 ChangeSpotStatus_Updates_Status	4 ms
 EditParkingLot_Updates_Record	3 ms
 GetAllParkingLot_Returns_Correct_Number	3 ms
 GetAvailableSpots_Service_Gets_Called	218 ms
 GetMessage_Service_Gets_Called	2 ms
 GetSpotStatus_Service_Gets_Called	8 ms
▲  ParkingSpotServiceTests (10)	1 sec
▲  GetAvailableSpots_Returns_Correct_Value (4)	46 ms
 GetAvailableSpots_Returns_Correct_Value(parkingLotId: 1, numberOfAvailable: 1)	3 ms
 GetAvailableSpots_Returns_Correct_Value(parkingLotId: 2, numberOfAvailable: 1)	38 ms
 GetAvailableSpots_Returns_Correct_Value(parkingLotId: 3, numberOfAvailable: 1)	2 ms
 GetAvailableSpots_Returns_Correct_Value(parkingLotId: 4, numberOfAvailable: 0)	3 ms
▲  IsSpotAvailable_Returns_Correct_Value (3)	10 ms
 IsSpotAvailable_Returns_Correct_Value(spotId: 1, isAvailable: False)	2 ms
 IsSpotAvailable_Returns_Correct_Value(spotId: 2, isAvailable: True)	6 ms
 IsSpotAvailable_Returns_Correct_Value(spotId: 3, isAvailable: False)	2 ms
▲  SelectSpot_Returns_Correct_Value (3)	954 ms
 SelectSpot_Returns_Correct_Value(parkingLotId: 1, daysFromNow: 0, expectedSpotId: 2)	21 ms
 SelectSpot_Returns_Correct_Value(parkingLotId: 1, daysFromNow: 1, expectedSpotId: 1)	927 ms
 SelectSpot_Returns_Correct_Value(parkingLotId: 2, daysFromNow: 0, expectedSpotId: 4)	6 ms
▲  ReservationControllerTests (4)	1,1 sec
 DeleteReservation_Deletes_Record	32 ms
 EditReservationStatus_Changes_Status_Correctly	27 ms
 GetAllReservations_Returns_Correct_Number	18 ms
 GetReservationById_Returns_Correct_Value	1 sec

13.1. ábra: A foglalások és a parkoló kezeléséért felelős controllerek, és a service tesztjei a Visual Studio Test Explorer nézetén

13.2. Kliens tesztek

A kliens teszteléséhez felhasználási eseteken vizsgáltam az alkalmazás működését. Ezt az alábbi táblázatban foglaltam össze:

Teszteteset sorszáma	Funkcionalitás	Elvárt eredmény	Teszt eredménye
1.	A regisztrációs felületen minden adatot megadva regisztráció az alkalmazásba.	Sikeres	Sikeres
2.	Létező felhasználóval belépés az alkalmazásba, sikeres belépés esetén átirányítás a főoldalra.	Sikeres	Sikeres
3.	Nem létező felhasználóval belépési kísérlet az alkalmazásba, hibaüzenetet tartalmazó ablak megjelenése.	Sikertelen	Sikertelen
4.	Egy, vagy több rendszám hozzáadása a felületen megadott kritériumnak megfelelően.	Sikeres	Sikeres

5.	Foglalás létrehozása még szabad helyekkel rendelkező parkolóba.	Sikeres	Sikeres
6.	Foglalás létrehozása ugyanazzal a rendszámmal, ugyanarra a napra.	Sikertelen	Sikertelen
7.	Foglalás módosítása másik napra és másik rendszámmal.	Sikeres	Sikeres
8.	Foglalás törlése még meg nem kezdett foglalás esetén.	Sikeres	Sikeres

13.2. táblázat: A kliens tesztelésének eredményei

13.3. ANPR tesztek

A rendszámfelismerő rendszer teszteléséhez a Plate Recognizer API-nak küldtem el POST kérésben az alábbi képeket:



13.3. ábra: Képek az automatikus rendszámfelismerés teszteléséhez

Ezeket a regisztrált felhasználói fiókomba belépve a Dashboard-on tudom megtekinteni:

License Plate	CL ?	Camera	Image
RVZ626	90.20%	-	
PPZ489	89.60%	-	
NLE003	90.10%	-	
RAP235	89.60%	-	
ASD123	90.10%	-	
ABC123	90.20%	-	

13.4. ábra: A Plate Recognizer Dashboard-ja

Jól látható, hogy a beküldött képeken szereplő rendszámokat megközelítőleg 90%-os valószínűséggel felismerte. A képeket a saját rendszeremből küldtem, az eredményeket ide kaptam vissza, tehát a képfelismerés és a rendszeremnek ez a funkciója működik.

✓ OCRTests (6)	25,6 sec
✓ GetLicensePlate_Returns_Correct_Value (6)	25,6 sec
✓ GetLicensePlate_Returns_Correct_Value(filePath: "OCR/asd123.jpg", expectedResult: "ASD123")	3,9 sec
✓ GetLicensePlate_Returns_Correct_Value(filePath: "OCR/nle003.jpg", expectedResult: "NLE003")	4,2 sec
✓ GetLicensePlate_Returns_Correct_Value(filePath: "OCR/ppz489.jpg", expectedResult: "PPZ489")	5,1 sec
✓ GetLicensePlate_Returns_Correct_Value(filePath: "OCR/rap235.jpg", expectedResult: "RAP235")	3,8 sec
✓ GetLicensePlate_Returns_Correct_Value(filePath: "OCR/rvz626.jpg", expectedResult: "RVZ626")	4,4 sec
✓ GetLicensePlate_Returns_Correct_Value(filePath: "OCR/vip.jpg", expectedResult: "ABC123")	4,2 sec

13.5. ábra: A rendszámfelismerés tesztjei a Visual Studio Test Explorer nézetén

13.4. Jelenlét érzékelés tesztek

Az ultrahangos szenzor teszteléséhez a mért eredményeket a soros monitorra íratom ki, itt fog látszódni, hogy mikor küldi az állapotváltozásokról szóló HTTP kéréseket és ezek sikerességéről ad visszajelzést. A parkolóban a hozzá tartozó helyen elhelyezett kijelzőn pedig frissül az üzenet. Egy helyet akkor érzékel foglaltnak, ha a mért távolság 50 cm alatt van, ha három egymást követő mérés eredménye ebbe az intervallumba esik, csak akkor küldi el a HTTP kérést. Ugyanígy, ha három egymást követő mérés eredménye legalább 50 cm akkor a hely szabadnak számít. Ezeket a kéréseket csak akkor küldi el, ha az előző állapothoz képest változás történik.



13.6. ábra: Ha nem érzékel a szenzor jelenlétet, ez a felirat jelenik meg

```
19:56:13.361 -> Connected to WiFi network with IP Address: 192.168.0.166
19:56:13.361 -> Distance: 98
19:56:13.361 -> Free counter set to 1
19:56:14.388 -> Distance: 98
19:56:14.388 -> Free counter set to 2
19:56:15.366 -> Distance: 99
19:56:15.366 -> Free counter set to 3
19:56:16.386 -> Distance: 97
19:56:16.386 -> Free counter set to 4
19:56:17.367 -> Distance: 98
19:56:17.367 -> Free counter set to 5
19:56:18.385 -> Distance: 97
19:56:18.385 -> Free counter set to 6
19:56:19.412 -> Distance: 98
19:56:19.412 -> Free counter set to 7
19:56:20.386 -> Distance: 99
19:56:20.386 -> Free counter set to 8
19:56:21.411 -> Distance: 99
19:56:21.411 -> Free counter set to 9
```

13.7. ábra: A mérések során egy számláló és az előző állapot segítségével tudjuk eldönteni, hogy el kell-e küldeni a kérést

FOGLALT

13.8. ábra: Ez a felirat jelenik meg, ha a szenzor jelenlétet érzékel

```
19:58:53.125 -> Distance: 97
19:58:53.125 -> Free counter set to 13
19:58:54.101 -> Distance: 12
19:58:54.101 -> Occupied counter set to: 1
19:58:55.126 -> Distance: 12
19:58:55.126 -> Occupied counter set to: 2
19:58:56.101 -> Distance: 12
19:58:56.101 -> Occupied counter set to: 3
19:58:57.126 -> Sending request with occupied status
19:58:57.173 -> HTTP Response code: 200
19:58:57.173 -> Distance: 12
19:58:57.173 -> Occupied counter set to: 1
19:58:58.197 -> Distance: 13
19:58:58.197 -> Occupied counter set to: 2
19:58:59.176 -> Distance: 13
19:58:59.176 -> Occupied counter set to: 3
19:59:00.200 -> Distance: 12
19:59:00.200 -> Occupied counter set to: 1
19:59:01.177 -> Distance: 12
19:59:01.177 -> Occupied counter set to: 2
```

13.9. ábra: Foglalt státusz elküldése

Ha a hely foglalt, de két egymás utáni mérést hibásan szabadnak érzékel, akkor a kérést nem fogja elküldeni:

```
20:03:18.038 -> Distance: 8
20:03:18.038 -> Occupied counter set to: 1
20:03:19.030 -> Distance: 7
20:03:19.030 -> Occupied counter set to: 2
20:03:20.021 -> Distance: 7
20:03:20.021 -> Occupied counter set to: 3
20:03:21.017 -> Sending request with occupied status
20:03:21.065 -> HTTP Response code: 200
20:03:21.065 -> Distance: 7
20:03:21.065 -> Occupied counter set to: 1
20:03:22.055 -> Distance: 8
20:03:22.055 -> Occupied counter set to: 2
20:03:23.056 -> Distance: 8
20:03:23.056 -> Occupied counter set to: 3
20:03:24.094 -> Distance: 8
20:03:24.094 -> Occupied counter set to: 1
20:03:25.087 -> Distance: 7
20:03:25.087 -> Occupied counter set to: 2
20:03:26.082 -> Distance: 10
20:03:26.082 -> Occupied counter set to: 3
20:03:27.067 -> Distance: 97
20:03:27.067 -> Free counter set to 1
20:03:28.107 -> Distance: 98
20:03:28.107 -> Free counter set to 2
20:03:29.101 -> Distance: 13
20:03:29.101 -> Occupied counter set to: 1
20:03:30.100 -> Distance: 8
20:03:30.100 -> Occupied counter set to: 2
20:03:31.094 -> Distance: 8
20:03:31.094 -> Occupied counter set to: 3
20:03:32.082 -> Distance: 7
20:03:32.082 -> Occupied counter set to: 1
```

13.10. ábra: Foglalt hely esetén, ha két mérés hibásan szabad eredményt ad, nem kerül sor új állapotváltásra

Ugyanígy, ha egy hely szabad, de két egymás utáni mérés tévesen foglalt, azt sem fogja elküldeni.

```
20:08:30.075 -> Distance: 98
20:08:30.075 -> Free counter set to 11
20:08:31.112 -> Distance: 99
20:08:31.112 -> Free counter set to 12
20:08:32.110 -> Distance: 8
20:08:32.110 -> Occupied counter set to: 1
20:08:33.103 -> Distance: 8
20:08:33.103 -> Occupied counter set to: 2
20:08:34.097 -> Distance: 99
20:08:34.097 -> Free counter set to 1
20:08:35.135 -> Distance: 98
20:08:35.135 -> Free counter set to 2
20:08:36.123 -> Distance: 100
20:08:36.123 -> Free counter set to 3
20:08:37.113 -> Distance: 100
20:08:37.113 -> Free counter set to 4
```

13.11. ábra: Szabad hely esetén, ha két mérés hibásan foglalt eredményt ad, nem kerül sor új állapotváltozásra

14. TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK

Az elkészült rendszer megfelel a feladateleírásban foglaltaknak, a vele szemben támasztott követelményeket teljesíti, azonban van lehetőség a továbbfejlesztésre.

Az elkészített alkalmazás bármilyen webes felületen elérhető böngészőből, azonban célszerű lenne hozzá mobilalkalmazás fejlesztése a felhasználói élmény fokozása érdekében.

Jelenleg a foglalás egy napig érvényes, így, ha valaki például csak délután érkezik a parkolóba, de már délelőtt lefoglalja a helyet, hogy biztosan ne teljen meg addigra a parkoló, akkor szükségtelen, hogy délelőtt is foglalt státusza legyen. Erre megoldást jelentene, ha a helyeket idősávokra is lehetne foglalni. Esetleg, ha valaki több napra szeretne magának biztos helyet, akkor ennek lehetővé tétele több napos foglalással. Előfordulhat, hogy amikor foglalni szeretne, éppen nincs elérhető hely, ekkor tudjon értesítést kérni az alkalmazástól, amint felszabadul egy hely, vagy akár ezt a helyet az alkalmazás automatikusan foglalja is le.

Amennyiben a felhasználók köre az egy szervezet (például az Egyetem) dolgozóira korlátozódik, akkor növelheti a felhasználói élményt, ha a belépést felhasználó kezelési szolgáltatás (például Active Directory) segítségével valósítjuk meg.

További fejlesztési lehetőség lehet egy fizetési megoldás implementálása is, ezzel további lehetséges ügyfelek előtt megnyitva a rendszer bevezetésének lehetőségét.

15.ÖSSZEFOGLALÓ

Szakdolgozatomban egy okos parkolórendszer fejlesztési folyamatához tekintettem át a kapcsolódó irodalmat, specifikáltam a feladatomat, követelményeket támasztottam a rendszerrel szemben. Ezek alapján megterveztem a saját rendszeremet, kiválasztottam a megvalósításához szükséges keretrendszereket, eszközöket és szoftvereket. A rendszert elkészítettem, a fejlesztés fontosabb lépéseit dokumentáltam. Az elkészült rendszert pedig teszteltem.

A rendszerem célja a parkolók esetén sokszor felmerülő problémákra megoldást találni. Problémát okozhat, hogy az autósok nem jutnak információhoz azzal kapcsolatosan, hogy mennyire telített egy parkoló, így ez tájékoztatás nélkül csak a helyszínen derül ki. Előfordulhat, hogy a parkolóban szükségük lenne egy számukra fenntartott helyre, melyet más nem vehet igénybe az adott időszakban. A parkoló üzemeltetője számára fontos lehet, hogy a parkolóba történő be-, és kilépések adminisztrálva legyenek.

A rendszerem fejlesztésénél szempont volt, hogy a felhasználó számára könnyen kezelhető webes felületen keresztül a parkolókból a helyfoglalást lehetővé tegyem. A helyszínen a beléptetés folyamatának megvalósítására olyan megoldásra volt szükség, amely nem vesz igénybe sok időt, nem tart tovább, mint a hagyományos beléptetés folyamata. A parkoló foglaltsági állapotáról valós idejű visszajelzést kell adni az alkalmazásban és a tájékoztató kijelzőkön is, ehhez a helyek megfigyelésére van szükség.

A fejlesztéshez példának vettem az Egyetemen működő belépőkártyás beléptetést, de a rendszer nem egy konkrét parkolóhoz készült, próbáltam egy általános megoldást találni, hogy a legtöbb parkolóhoz könnyen lehessen implementálni, bevezetése ne járjon nagyobb átalakításokkal az eredeti rendszerben.

Az elkészült rendszer teljesíti a követelmény specifikációban foglaltakat, de további funkciókkal való bővítésére, fejlesztésére van lehetőség, ezeket a Továbbfejlesztési lehetőségek fejezetben foglaltam össze.

16. SUMMARY

In my thesis work I researched the relevant literature for the development process of a smart parking system. I have specified my tasks and defined the system requirements. After that I created the system design, selected the needed frameworks, tools and softwares for the implementation. I have documented the most important steps of the development process. Finally, I have tested the different components of the system.

The aim of the parking system is to find solution to the problems that may occur during parking. In many cases, drivers do not know about the availability of a parking lot, and this information is only revealed after arriving to the parking lot, which results in many wasted minutes for finding a new spot. It is possible that we need a reserved parking spot, which is not available for other people in the reserved time frame. The administration of entries and exits from the parking lot can also be important to the owner or operator of the parking lot.

It was an important aspect of development, that the user should be able to reserve a spot in the parking lot with the help of an easy-to-use web application. The entry process had to be implemented on a way that does not take too much time, not longer than the normal entry process. The application must give real time feedback to the users about the availability of the parking lot, both in the application and on-site information screen. In order to fulfill this requirement, parking spots need to be monitored constantly.

Although I used the University's parking system as an example, which currently operates with an access card, my system is not designed to just one specific parking lot. I tried to find a more general solution, which can be used in other parking lots without major modifications.

The completed system fulfills the specified requirements, but there is room for the further development of additional functions. These possible development ideas have also been analyzed in my thesis work.

17. IRODALOMJEGYZÉK

[1] H. Arasteh et al., "Iot-based smart cities: A survey," 2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC), 2016, pp. 1-6, doi: 10.1109/EEEIC.2016.7555867.

(<https://ieeexplore.ieee.org/abstract/document/7555867>) utoljára megtekintve: 2021.10.27.

[2] M.Y.I. Idris, Y.Y. Leng, E.M. Tamil, N.M. Noor and Z. Razak: Car Park System: A Review of Smart Parking System and its Technology, Faculty of Computer Science and Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia

(https://d1wqtxts1xzle7.cloudfront.net/48697684/Car_Park_System_A_Review_of_Smart_Parkin20160909-9520-12i3bz6-with-cover-page-v2.pdf?Expires=1635377021&Signature=UVwBdXLJfQ~QmmnsKo7yGRqvTkqj7LKOr9JxXwSWFApxtS1uHOSaTB0mFENQCdvJMgXs5I3i0VnPf-DcN2t2nT2Dn4U5mK05n5r1ehZAFIOxYgutJNgoJM8peaEw6tHyfP5CHSqO22Z5jjELPrSN3--YbLItKMBrEzLmqo2yst0Sui0LnB7o-QKDv8fVKLdzsFY8qHm6gDseQlsCITvs5mDWtrEwX0BY95z4v8dm6Ebd4EwqK3TQjh9WSep1Qd3NCOjMFbpJI4lzsK7Rd6czsPRqlT7rUJjUFxHBONmPN3Y4na0nbNV0ArIcq5nXnWC0ATz2oUZkAeth9l9kGT26wg_&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA), utoljára megtekintve: 2021. 10. 27.

[3] Parking Guidance

(<https://www.hubparking.com/features/parking-guidance/>), utoljára megtekintve: 2021.10.28.

[4] Mikrohullámú radar

<https://oktel.hu/szolgalatas/riasztobereendezesek/mozgaserzekelok/radaros-infra/>, utoljára megtekintve: 2021.11.01.

[5] Passzív infravörös szenzorok

<https://www.arrow.com/en/research-and-events/articles/understanding-active-and-passive-infrared-sensors>

utoljára megtekintve: 2021. 11. 01.

[6] RFID

<https://smartfreq.hu/az-rfid-technologia-elemei-es-mukodese/>, utoljára megtekintve: 2021. 11. 01.

[7] Lovas István, Ládi Alexander:

(https://elearning.uni-obuda.hu/main_archive/pluginfile.php/760218/mod_resource/content/2/07_Szenzorok_4.pdf) utoljára megtekintve: 2021. 11. 01.

[8] Design of real time vehicle identification system

(<https://ieeexplore.ieee.org/abstract/document/400182>), utoljára megtekintve: 2021. 11. 01.

[9] License Plate Recognition

(<https://survisiongroup.com/post-what-is-license-plate-recognition>), utoljára megtekintve: 2021. 11. 03.

[10] Plate Recognizer

(<https://platerecognizer.com/>), utoljára megtekintve: 2021. 11. 06.

[11] Tesseract OCR

(<https://bytepace.medium.com/what-is-tesseract-and-how-it-works-dfff720f4a32>), utoljára megtekintve: 2021. 11. 07.

[12] Rekor's OpenALPR

(<https://www.openalpr.com/>), utoljára megtekintve: 2021. 11. 08.

[13] MVC

(https://docs.microsoft.com/hu-hu/aspnet/core/mvc/overview?WT.mc_id=dotnet-35129-website&view=aspnetcore-6.0), utoljára megtekintve: 2021. 11. 14.

[14] MVVM

(<https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm>), utoljára megtekintve: 2021. 11. 14.

[15] MVC, MVVM ábrák

(https://users.nik.uni-obuda.hu/prog4/Eloadas_HU/P4HU_EA_01_GuiIntro.pptx), utoljára megtekintve: 2021. 11. 14.

[16] ASP.NET Web API

(<https://www.tutorialsteacher.com/webapi/what-is-web-api>), utoljára megtekintve: 2021. 11. 14.

[17] Angular

(<https://hackr.io/blog/why-should-you-learn-angular>), utoljára megtekintve: 2021. 11. 14.

- [18] Adatbázis-motorok rangsora népszerűségük alapján
(<https://db-engines.com/en/ranking>), utoljára megtekintve: 2022. 04. 17.
- [19] SQLite
(<https://www.sqlite.org/index.html>), utoljára megtekintve: 2022. 04. 17.
- [20] REST architektúrális stílus
(<https://www.codecademy.com/article/what-is-rest>), utoljára megtekintve: 2022. 04. 17.
- [21] A kliens és a szerver kapcsolata
(<https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-6.0&tabs=visual-studio>), utoljára megtekintve: 2022. 04. 17.
- [22] Angular
(<https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-angular-development/>), utoljára megtekintve: 2022. 04. 25.
- [23] JSON Web Tokens
(<https://jwt.io/introduction>), utoljára megtekintve: 2022. 05. 02.
- [24] Arrange-Act-Assert
(<https://automationpanda.com/2020/07/07/arrange-act-assert-a-pattern-for-writing-good-tests/>), utoljára megtekintve: 2022. 05. 08.
- [25] xUnit.net
(<https://xunit.net/>), utoljára megtekintve: 2022. 05. 08.
- [26] The different types of software testing
(<https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing>), utoljára megtekintve: 2022. 05. 08.

18.ÁBRAJEGYZÉK

1.1. ábra: Az okos városok alkotó egységei [1].....	1
2.1. ábra: Beltéri PGIS	3
4.1. ábra: A tervezett rendszer főbb folyamatainak ismertetése	6
5.1. ábra: Mikrohullámú radaros mozgásérzékelő modul.....	7
5.2. ábra: Digitális akusztikus érzékelő	8
5.3. ábra: RFID kártya író és olvasó modul	9
5.4. ábra: Ultrahangos távolságérzékelő modul	9
6.1. ábra: Koreai rendszám	11
6.2. ábra: Snapshot és Stream összehasonlítása.....	13
6.3. ábra: Scout és CarCheck összehasonlítása.....	14
7.1. ábra: MVC architekturális minta [15].....	15
7.2. ábra: MVVM architekturális minta [15].....	16
8.1. ábra: Saját rendszer terve	17
9.1. ábra: Use Case diagram a felhasználói felület funkcióihoz (Az ábra az app.diagrams.net alkalmazással készült)	19
10.1. ábra: Adatbázis motorok rangsora népszerűségük alapján [18]	22
10.2. ábra: Kommunikáció a kliens és a szerver között [21].....	23
10.3. ábra: Regisztrációs felület.....	24
10.4. ábra: Bejelentkezési képernyő	25
10.5. ábra: Főoldal	25
10.6. ábra: Foglалás létrehozása	26
10.7. ábra: Autók rendszámainak felvétele.....	26
10.8. ábra: Foglалások megtekintése	27
10.9. ábra: Admin felület	27
10.10. ábra: HC-SR04-4P ultrahangos távolságérzékelő modul	28
10.11. ábra: NodeMCU-ESP8266-CH.....	28
11.1. ábra: A saját rendszeremet alkotó komponensek és a köztük lévő adatátviteli kapcsolatok (Az ábra az app.diagrams.net alkalmazással készült).....	29
12.1. ábra: Az adatbázis egyed-kapcsolat diagramja (Az ábra az app.diagrams.net alkalmazással készült).....	32
12.2. ábra: A főoldal belépés előtt	38
12.3. ábra: Regisztrációs felület.....	38
12.4. ábra: Bejelentkezési képernyő	38
12.5. ábra: Sikeres bejelentkezés	39
12.6. ábra: Főoldal bejelentkezett felhasználók számára.....	39
12.7. ábra: Foglалás létrehozása	40
12.8. ábra: Visszajelzés a foglalás sikerességéről.....	40
12.9. ábra: A bejelentkezett felhasználó foglalásai.....	41
12.10. ábra: Foglалás módosítása	41
12.11. ábra: A megjelenő üzenet, ha nem sikerül a foglalás	42

12.12. ábra: Üzenet a foglalás törlésének sikerességéről.....	42
12.13. ábra: Rendszám felvétele	43
12.14. ábra: Admin felületen megjelenik a foglalt státuszú foglalás.....	43
12.15. ábra: Az admin a „Megérkezett” gombbal „Folyamatban” státuszúra tud állítani egy foglalást	43
12.16. ábra: A Plate Recognizer által visszaadott eredmény alapján a beléptetés folyamatábrája (Az ábra az app.diagrams.net alkalmazással készült).....	45
12.17. ábra: Az echo és trig pin-ek a D5 és D4 pin-ekhez csatlakoznak.....	46
12.18. ábra: A szenzort 5 V feszültséghez és a GND-hez van csatlakoztatva.....	46
12.19. ábra: Az ultrahangos szenzorhoz csatlakozó kábelek.....	47
12.20. ábra: A parkolóban vannak szabad helyek	47
12.21. ábra: A parkolóban nincs már szabad hely	48
12.22. ábra: Az adott rendszámmal foglaltak helyet az alkalmazásban, ennek a számát jeleníti meg	48
12.23. ábra: A fenntartott hellyel rendelkező személy foglalás nélkül beléphet	48
12.24. ábra: A rendszámmal nem foglaltak helyet, de még van szabad hely a parkolóban, ezért beléphet	49
12.25. ábra: Az érkező autósna nincs foglalása, nem rendelkezik fenntartott hellyel és a parkolóban nincs már szabad hely, így nem léphet be	49
12.26. ábra: A hely tulajdonosa különleges jogkörrel rendelkező személy.....	49
12.27. ábra: A kijelzőn "Foglalt" szöveg jelenik meg, ha az autós foglalás nélkül érkezett és beállt az egyik helyre.....	50
12.28. ábra: Az alkalmazásban foglalást beküldő autós helye	50
12.29. ábra: A hely szabad.....	50
13.1. ábra: A foglalások és a parkoló kezeléséért felelős controllerek, és a service tesztjei a Visual Studio Test Explorer nézetén.....	52
13.2. táblázat: A kliens tesztelésének eredményei.....	53
13.3. ábra: Képek az automatikus rendszámfelismerés teszteléséhez	53
13.4. ábra: A Plate Recognizer Dashboard-ja.....	54
13.5. ábra: A rendszámfelismerés tesztjei a Visual Studio Test Explorer nézetén.....	54
13.6. ábra: Ha nem érzékel a szenzor jelenlétet, ez a felirat jelenik meg	55
13.7. ábra: A mérések során egy számláló és az előző állapot segítségével tudjuk eldönteni, hogy el kell-e küldeni a kérést	55
13.8. ábra: Ez a felirat jelenik meg, ha a szenzor jelenlétet érzékel	56
13.9. ábra: Foglalt státusz elküldése	56
13.10. ábra: Foglalt hely esetén, ha két mérés hibásan szabad eredményt ad, nem kerül sor új állapotváltozásra	57
13.11. ábra: Szabad hely esetén, ha két mérés hibásan foglalt eredményt ad, nem kerül sor új állapotváltozásra	58

19.TÁBLÁZATJEGYZÉK

5.1. táblázat: Példák intruzív és nem intruzív szenzorokra.....	7
12.1. táblázat: Az autentikációért felelős controller metódusai.....	33
12.2. táblázat: A foglalások kezeléséért felelős controller metódusai.....	35
12.3. táblázat: A parkoló adatainak kezeléséért felelős controller metódusai	36
13.2. táblázat: A kliens tesztelésének eredményei.....	52