

# Parkolóház Kezelő Rendszer

Pannonhalmi Főapátság

Szegedi SOB Technikuma



## Szakképesítés

Szoftverfejlesztő és -tesztelő

### Témavezető:

Rédai Dávid

### Készítettek

Juhász Szabolcs

Scher János

Major Attila

2025

Szeged

# Tartalom

---

A projekt áttekintése .....	3
A projekt céljának részletes ismertetése .....	4
Tervezési szempontok és alkalmazott módszerek .....	5
SWOT elemzés .....	9
Feladatmegosztás a projektcsapaton belül .....	10
Ütemterv – a projekt előrehaladásának menete .....	12
Felhasználói felület és alkalmazott technológiák .....	15
1. Bejelentkezés és Regisztráció .....	15
2. Járműkezelés .....	16
3. Parkolási Események .....	17
4. Admin Panel .....	19
5. Mobil applikáció .....	21
Adatbázisok és eszközök részletes bemutatása .....	23
Fejlesztői Környezet .....	25
1. Backend .....	25
2. Frontend .....	25
3. Adatbázis .....	25
4. Deployment .....	26
5. Verziókezelés .....	28
Minőségbiztosítás és tesztelési folyamatok .....	29
API Tesztelés .....	29
Frontend Tesztelés .....	34
Környezeti Tesztelés .....	37
Biztonsági Tesztelés .....	38
Összegzés .....	39
Felhasznált források .....	40
Ábrajegyzék .....	41

## A projekt áttekintése

---

A **Parkolóház Kezelő Rendszer** célja egy olyan korszerű, digitális szoftvermegoldás létrehozása volt, amely lehetővé teszi egy modern parkolóház hatékony, automatikus és felhasználóbarát üzemeltetését. A rendszer képes kezelni a felhasználók regisztrációját, a járművek nyilvántartását, a parkolási események rögzítését, valamint a parkolási díjak kiszámítását és megjelenítését. Emellett egy adminisztrációs felületet is biztosít az üzemeltetők számára, amelyen keresztül teljes körű menedzsment végezhető.

A rendszer modern webes felülete (Svelte) és mobilalkalmazása lehetővé teszi a felhasználók számára a parkolóhelyek foglalását, a járművek kezelését és a parkolási előzmények nyomon követését. A backend (ASP.NET Core 8.0) biztosítja a biztonságos adatkezelést, a valós idejű kommunikációt és a komplex üzleti logikák feldolgozását. A rendszer automatikusan generál PDF számlákat, küld email értesítéseket, és részletes statisztikákat készít a parkolóház használatáról.

## A projekt céljának részletes ismertetése

---

A fejlesztés célja egy **komplex, modulárisan felépített parkoloház-kezelő rendszer** kialakítása volt, amely képes kiszolgálni különböző felhasználói igényeket, és hosszú távon is skálázható marad. A rendszer főbb funkciói közé tartozik:

- **Felhasználói fiókok kezelése:** regisztráció, bejelentkezés, jogosultságok kezelése.
- **Járműregisztráció:** a felhasználók saját járműveiket rögzíthetik a rendszerben.
- **Parkolási események nyilvántartása:** a parkolóba történő be- és kihajtások naplózása, időbélyeggel.
- **Díjszámítás:** a parkolási idő és a díjtételek alapján automatikus díjszámítás történik.

A rendszer többplatformos elérést biztosít: **webes és mobilfelületet** is kínál, így a felhasználók bármilyen eszközről kényelmesen hozzáférhetnek. A fejlesztés során kiemelt figyelmet fordítottunk az alábbi szempontokra:

- **Modularitás:** a rendszer könnyen bővíthető további funkciókkal.
- **Skálázhatóság:** alkalmas nagyobb parkoloházak kiszolgálására is.
- **Információbiztonság:** biztonságos jelszókezelés, szerepköralapú hozzáférés, adatok titkosítása.

## Tervezési szempontok és alkalmazott módszerek

---

A rendszer fejlesztését egy előre kidolgozott, **20 hetes fejlesztési terv** mentén végeztük, amely lefedte az igényfelméréstől kezdve a tesztelésen át a dokumentáció elkészítéséig minden fontosabb fazist. A projekt során különös figyelmet fordítottunk a korszerű technológiák és fejlesztési irányelvek alkalmazására, amelyek biztosítják a hosszú távú fenntarthatóságot és a további fejlesztések lehetőségét.

A backend oldal fejlesztéséhez a **.NET 8** platformot választottuk, amely megbízható alapot biztosít a REST API-k kialakításához. Az **Entity Framework Core** ORM használatával elkerültük a manuális SQL lekérdezések írását, miközben megőriztük a teljes kontrollt az adatbázis műveletek felett:



```
C# Program.cs M X
C# Program.cs
46 |
47 // Add services to the container.
48 builder.Services.AddDbContext<ApplicationDbContext>(options =>
49     options.UseMySql(connectionString,
50         new MySqlServerVersion(new Version(8, 0, 13)),
51         mySqlOptions => mySqlOptions.EnableRetryOnFailure()));
```

1.ábra: Entity Framework Core és MySQL adatbáziskapcsolat konfigurációja

A rendszer aszinkron feldolgozást és strukturált hibakezelést alkalmaz, ezzel növelve a teljesítményt és stabilitást:

```
C# ReservationController.cs 4 ×
Controller > ReservationController > C# ReservationController.cs > GetMyReservations
13 public class ReservationController : ControllerBase
90
91     // Felhasználó saját foglalásainak lekérdezése
92     [HttpGet("my")]
93     public async Task<IActionResult> GetMyReservations()
94     {
95         try
96         {
97             var userEmail = User.FindFirstValue(ClaimTypes.Name);
98             var user = await _context.Users.FirstOrDefaultAsync(u => u.Email == userEmail);
99
100            if (user == null)
101                return NotFound("Felhasználó nem található");
102
103            var reservations = await _context.Reservations
104                .Where(r => r.UserId == user.Id)
105                .Include(r => r.ParkingSpot)
106                .OrderByDescending(r => r.StartTime)
107                .Select(r => new
108                {
109                    id = r.Id,
110                    startTime = r.StartTime,
111                    endTime = r.EndTime,
112                    status = r.Status,
113                    totalFee = r.TotalFee,
114                    floorNumber = r.ParkingSpot.FloorNumber,
115                    spotNumber = r.ParkingSpot.SpotNumber
116                })
117                .ToListAsync();
118
119            return Ok(reservations);
120        }
121    }
```

2. ábra: Felhasználói foglalások lekérdezése a ReservationController-ben

A frontend technológiáit a **Svelte** képezi, amely a reaktív működésének köszönhetően gyors és letisztult felhasználói élményt nyújt. A keretrendszer használata lehetővé tette a dinamikus komponensek gyors fejlesztését és egyszerű karbantartását.

Az adatkezelés központi eleme egy **MySQL alapú adatbázis**, amelyet az **Aiven felhőszolgáltatás** segítségével hosztoltunk. Ez lehetővé tette a könnyű skálázást, valamint a megbízható és biztonságos elérést különböző környezetekből:



```
Program.cs M X
Program.cs
18
19 // MySQL kapcsolat
20 var host = Environment.GetEnvironmentVariable("MYSQL_HOST");
21 var port = Environment.GetEnvironmentVariable("MYSQL_PORT");
22 var database = Environment.GetEnvironmentVariable("MYSQL_DATABASE");
23 var user = Environment.GetEnvironmentVariable("MYSQL_USER");
24 var password = Environment.GetEnvironmentVariable("MYSQL_PASSWORD");
25 var sslMode = Environment.GetEnvironmentVariable("MYSQL_SSL_MODE") ?? "REQUIRED";
26
27 // Ellenőrizzük és naplózzuk a környezeti változókat
28 Console.WriteLine($"MYSQL_HOST: {host}");
29 Console.WriteLine($"MYSQL_PORT: {port}");
30 Console.WriteLine($"MYSQL_DATABASE: {database}");
31 Console.WriteLine($"MYSQL_USER: {user}");
32 Console.WriteLine($"MYSQL_SSL_MODE: {sslMode}");
33
34 if (string.IsNullOrEmpty(host) || string.IsNullOrEmpty(port) || string.IsNullOrEmpty(database) ||
35     string.IsNullOrEmpty(user) || string.IsNullOrEmpty(password))
36 {
37     throw new Exception("Missing required environment variables for database connection");
38 }
39
40 var connectionString = $"Server={host};" +
41     $"Port={port};" +
42     $"Database={database};" +
43     $"User={user};" +
44     $"Password={password};" +
45     $"SslMode={sslMode}";
```

3. ábra: MySQL kapcsolat környezeti változókkal

A jelenlegi megvalósításban a felhasználi jelszavak **Base64 kódolással** kerülnek eltárolásra, amely elsősorban a karakterek átalakítására szolgál, de nem nyújt valódi védelmet. Ennek tudatában a jövőbeni fejlesztési terv részeként szerepel egy **kriptográfialag biztonságos, egyirányú hash-elési algoritmus** – például a **bcrypt** – bevezetése, amely jelentősen növelné a rendszer adatbiztonságát, különösen a jelszavak kezelése terén:

```
C# UsersController.cs 4 ×
Controller > UsersController.cs > UsersController > .ctor
17 public class UsersController(ApplicationContext context) : ControllerBase
20     public IActionResult Register([FromBody] User user)
40
41         user.PasswordHash = Convert.ToBase64String(Encoding.UTF8.GetBytes(user.PasswordHash));
42         context.Users.Add(user);
43         context.SaveChanges();
44         return Ok("User registered successfully.");
45
```

4. ábra: Jelszavak Base64-es kódolása

A fejlesztés során alkalmazott tervezési alapelvek:

- **Szeparált rétegstruktúra:** a backend, frontend és adatbázis külön komponensként lett meghatározva, ami elősegíti a kód újra felhasználhatóságát és a hibakeresés egyszerűségét.
- **Verziókezelés:** a projekt teljes fejlesztése **Git** alapon zajlott, a GitHub-on való együttműködés mellett.
- **Kódolási szabványok:** betartottuk a **C#** és **JavaScript** nyelvi konvencióit, biztosítva az olvashatóságot és a karbantarthatóságot.
- **Tesztelhetőség:** a rendszer logikájának kialakítása során ügyeltünk az egységesítés lehetőségeire, amely hosszú távon hozzájárul a minőségbiztosításhoz.

## SWOT elemzés

---

A projekt elemzése során **SWOT-analízist** alkalmaztunk, amely egy széles körben használt marketing és üzleti tervezési technika. Lehetővé teszi, hogy a rendszer **belső tényezőit** (erősségek, gyengeségek) és **külső környezeti hatásait** (lehetőségek, veszélyek) átlátható módon vizsgáljuk. Az elemzés az alábbi kategóriák mentén történt:

- **S – Strengths (Erősségek)**
- **W – Weaknesses (Gyengeségek)**
- **O – Opportunities (Lehetőségek)**
- **T – Threats (Veszélyek)**

	<u>Belső tényezők</u>	<u>Külső tényezők</u>
<b><u>Pozitív tényezők</u></b>	<ul style="list-style-type: none"><li>+ Letisztult, reszponzív és felhasználóbarát kezelőfelület</li><li>+ Jól strukturált, moduláris backend architektúra</li><li>+ Platformfüggetlen elérés (webes és mobilfelületen is használható)</li></ul>	<ul style="list-style-type: none"><li>+ Fizetési rendszer integrálása (pl. Stripe, PayPal)</li><li>+ Valós idejű parkolohely-információ és foglalási lehetőség</li><li>+ QR-kódos be- és kiléptetés megvalósítása</li></ul>
<b><u>Negatív tényezők</u></b>	<ul style="list-style-type: none"><li>- A fizetési rendszer integrációja jelenleg még nem valósult meg</li><li>- Internetkapcsolat szükséges a rendszer használatához</li><li>- A mobilalkalmazás funkciói még korlátozottak</li></ul>	<ul style="list-style-type: none"><li>- Adatvédelmi előírások megsértésének kockázata hibás fejlesztés esetén</li><li>- Adatvesztés vagy rendszerleállás lehetősége nem megfelelő mentési stratégia mellett</li></ul>

5. ábra: SWOT analízis

## Feladatmegosztás a projektcsapaton belül

---

A projektet egy háromfős csapatban készítettük, és mindenki a saját szakterületének megfelelően járult hozzá a munka sikeréhez. Az alábbiakban összegzem, hogyan osztottuk el a feladatokat.

### **Juhász Szabolcs – Frontend fejlesztés, Tesztelés, Csapatvezetés**

- Szabolcs volt felelős a felhasználói felület kialakításáért, hogy az könnyen használható és reszponzív legyen. A frontend fejlesztése mellett a tesztelésben is részt vett, hogy biztosítsa, hogy az alkalmazás minden platformon jól működjön. Ő irányította a csapatot is, hogy a projekt zökkenőmentesen haladjon.

### **Scher János – Backend fejlesztés, Mobilalkalmazás fejlesztés**

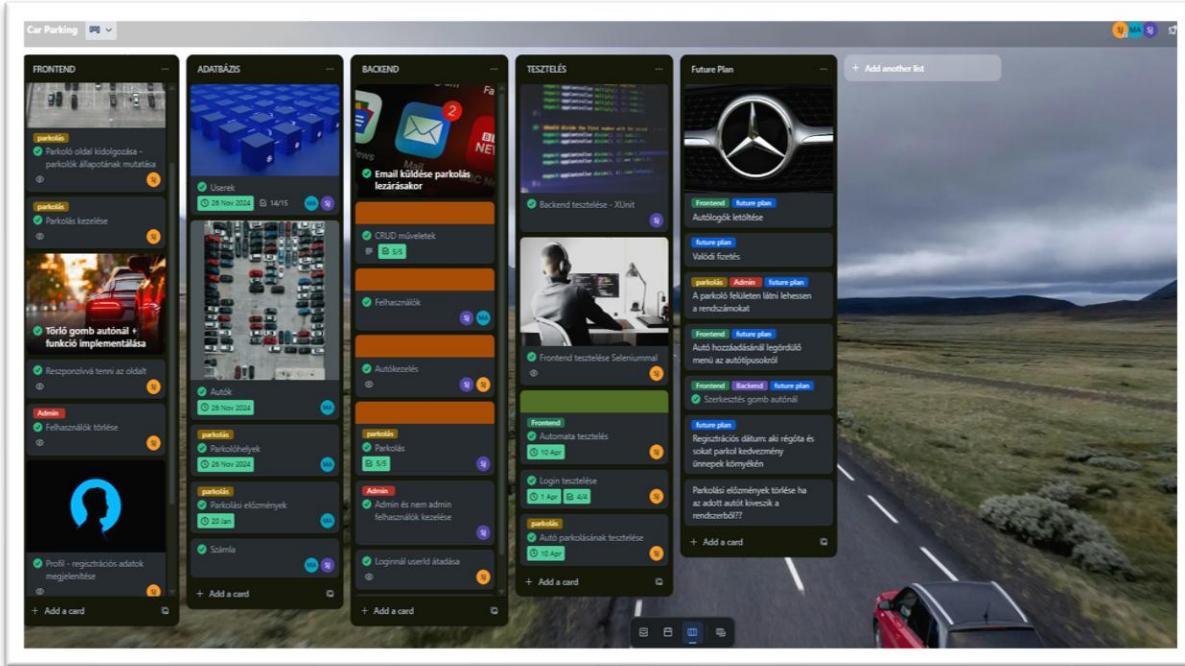
- János volt felelős a backend fejlesztéséért, amely ASP.NET Core technológiával készült. A rendszer logikáját, adatkezelését és biztonságát ő biztosította. Ezen kívül dolgozott a mobilalkalmazás fejlesztésén is, hogy a felhasználók okostelefonon is könnyelmesen használhassák a parkolóház rendszert.

### **Major Attila – Adatbázis tervezés, Dokumentáció készítés**

- Attila felelt az adatbázis tervezéséért, hogy az jól strukturált és biztonságos legyen. Az ő feladata volt a dokumentáció készítése is, amely segített abban, hogy a projekt átlátható és könnyen karbantartható legyen a jövőben.

### **Együttműködés és Kommunikáció**

A csapat minden tagja szorosan együttműködött, és rendszeresen kommunikáltunk, hogy minden feladat a megfelelő ütemben haladjon. A projekt különböző részei, mint a frontend, backend és mobilalkalmazás, folyamatos egyeztetés mellett készültek el. A célunk az volt, hogy egy jól működő és átfogó rendszert hozzunk létre, amely a parkolóházak hatékony kezelését segíti. A Trello ingyenes verzióját használtuk a feladatok nyomon követésére:



6. ábra: Trello

Ez a jól szervezett feladatmegosztás segített abban, hogy a projekt sikeresen megvalósuljon, és az elkészült parkolóház-kezelő rendszer megfeleljen a vizsgamunka követelményeinek.

# Ütemterv – a projekt előrehaladásának menete

---

A projekt fejlesztését 20 hetes ciklusban végeztük. A heti ütemezés az alábbiak szerint történt:

## 1. Hét: Projektindítás és Analízis

A projekt első hetében sor került a csapat összeállítására, a szerepkörök meghatározására, valamint a részletes követelménygyűjtésre a potenciális felhasználókkal. Piackutatás és konkurenciaelemzés is készült, majd kiválasztottuk a szükséges technológiákat. Végül beállítottuk a fejlesztői környezetet, és bevezettük a projektmenedzsment eszközöket (Trello).

## 2. Hét: Adatbázis Tervezés

Elkészítettük a részletes ER diagramot, elvégeztük az adatbázis normalizálását, valamint kidolgoztuk a teljesítmény-optimalizálási terveket. Meghatároztuk a migrációs stratégiákat, valamint elkészítettük a backup/recovery és biztonsági terveket is.- Backup és recovery terv kidolgozása

## 3. Hét: Backend Architektúra

Létrehoztuk az ASP.NET Core 8.0 projektstruktúrát, beállítottuk az Entity Framework Core-t és a MySQL adatbáziskapcsolatot. Megterveztük az API végpontokat, konfiguráltuk a Swagger dokumentációt, valamint a környezeti változók kezelését.

## 4. Hét: Frontend Architektúra

Inicializáltuk a Svelte projektet. Megterveztük a komponens architektúrát és kialakítottuk az API integrációs réteget.

## 5. Hét: Felhasználói Autentikáció

Implementáltuk a cookie alapú hitelesítést, létrehoztuk a regisztrációs és bejelentkezési folyamatokat, valamint az email értesítési rendszert.

## **6. Hét: Járműkezelés Rendszer**

Létrehoztuk az Car entitást, implementáltuk a CRUD műveleteket, hozzáadtuk az automatikus logóbetöltést, validáltuk az adatokat, valamint elkészítettük a listázási, szűrési és törlési funkciókat.

## **7. Hét: Parkolás Kezelés**

Létrehoztuk a ParkingSpot entitást, kialakítottuk a parkolás indítási/befejezési funkciókat, díjszámítást, foglalási lehetőséget, vizuális megjelenítést és a foglalt/szabad státusz jelzést.

## **8. Hét: Adminisztrációs Rendszer**

Implementáltuk az admin jogosultságkezelést, a felhasználók és parkolóhelyek kezelését, valamint a statisztikák megtekintésének lehetőségét.

## **9. Hét: Számlázási Rendszer**

Integráltuk az iTextSharp könyvtárat, lehetővé tettük PDF számlák generálását, letöltését, újraküldését, stáruszuk kezelését és az előzmények megtekintését.

## **10. Hét: Statisztikák és Rijportok**

Kialakítottuk a parkolási előzmények, összesítők, autónkenti és havi statisztikák, bevételi kimutatások és felhasználói aktivitások megjelenítését.

## **11. Hét: Email Rendszer**

Integráltuk a MailKit könyvtárat, valamint létrehoztuk a számlák emailes kiküldésének rendszerét.

## **12. Hét: Frontend Fejlesztés**

Tovább fejlesztettük a felhasználói felületet: reszponzivitás, vizuális komponensek, állapotkezelés, hibakezelés, betöltési állapotok és visszajelzések kerültek beépítésre.

## **13. Hét: API Fejlesztés**

Megvalósítottuk a végpontokat, validációkat, hibakezeléseket, teljesítmény-optimalizálást, biztonsági ellenőrzéseket és frissítettük az API dokumentációt.

## **14. Hét: Tesztelés**

Elvégeztük a kézi Swagger tesztelést az API funkciók ellenőrzéséhez.

## **15. Hét: Dokumentáció**

Elkészült az API dokumentáció, a telepítési útmutató, valamint a kódhoz kapcsolódó kommentek és leírások.

## **16. Hét: Deployment**

Konfiguráltuk a Render és Netlify szolgáltatásokat, beállítottuk az Aiven MySQL kapcsolatot, a környezeti változókat, SSL/TLS-t és CORS-t.

## **17. Hét: Teljesítmény Optimalizálás**

Optimalizáltuk a backend és frontend teljesítményét.

## **18. Hét: Biztonság**

Adatvédelmi fejlesztéseket végeztünk.

## **19. Hét: Stabilizálás**

Elvégeztük a hibakezelést, naplózást, session kezelést, adatbázis stabilizálást, valamint a számlázási rendszer, frontend és biztonság végső simításait.

## **20. Hét: Projekt Zárás**

Végső tesztelés, dokumentáció frissítés, valamint a projekt bemutatójához szükséges prezentáció (ppt) elkészítése történt meg.

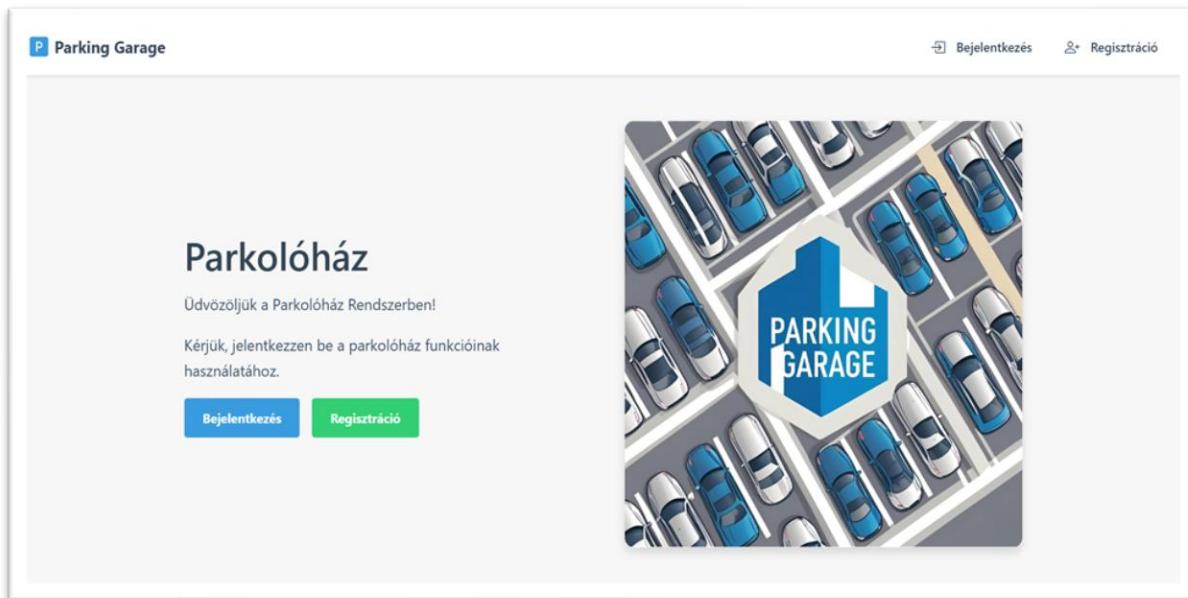
A projekt során a heti ütemezés szerint haladtunk, és minden modul fejlesztése összhangban történt az előre megtervezett lépésekkel. A backend ASP.NET Core 8 technológiával, a frontend Svelte alapokon készült. A fejlesztés során külön figyelmet fordítottunk a rezponzív felhasználói felületre, valamint a megbízható adatkezelésre.

## Felhasználói felület és alkalmazott technológiák

A felhasználói felület a Svelte keretrendszer segítségével készült. A rendszer reszponzív, azaz asztali és mobil eszközökön egyaránt használható. A dizájn célja az egyszerű kezelhetőség és átláthatóság biztosítása volt, modern, letisztult megjelenéssel. A frontend főbb oldalai és funkciói:

### 1. Bejelentkezés és Regisztráció

- Felhasználói regisztráció email cím és jelszó megadásával
- Bejelentkezés a regisztrált felhasználói adatokkal
- Biztonságos kijelentkezés



7. ábra: Főoldal

## 2. Járműkezelés

- Új jármű hozzáadása rendszám és márka megadásával
- Járművek listázása és kezelése
- Automatikus márka logók betöltése
- Jármű adatok szerkesztése és törlése

Parking Garage

Főoldal Autóim Előzmények Parkoló Profil Kijelentkezés

## Autóim

+ Autó hozzáadása

Márka	Modell	Rendszám	Évjárat	Parkolási állapot	Gombok
Opel	Astra	RAW-281	2011	Jelenleg parkolva	Parkolás leállítása Törés
Ford	Mustang GT	FORD-007	1974	Nincs parkolva	Parkolás indítása Törés

8. ábra: Saját autók megjelenítése

Főoldal Autóim Előzmények

### Új autó hozzáadása

Márka\*

pl. Toyota

Modell\*

pl. Corolla

Évjárat

2025

Rendszám\*

pl. ABC-123

Mégse Mentés

Parkoloház © 2025

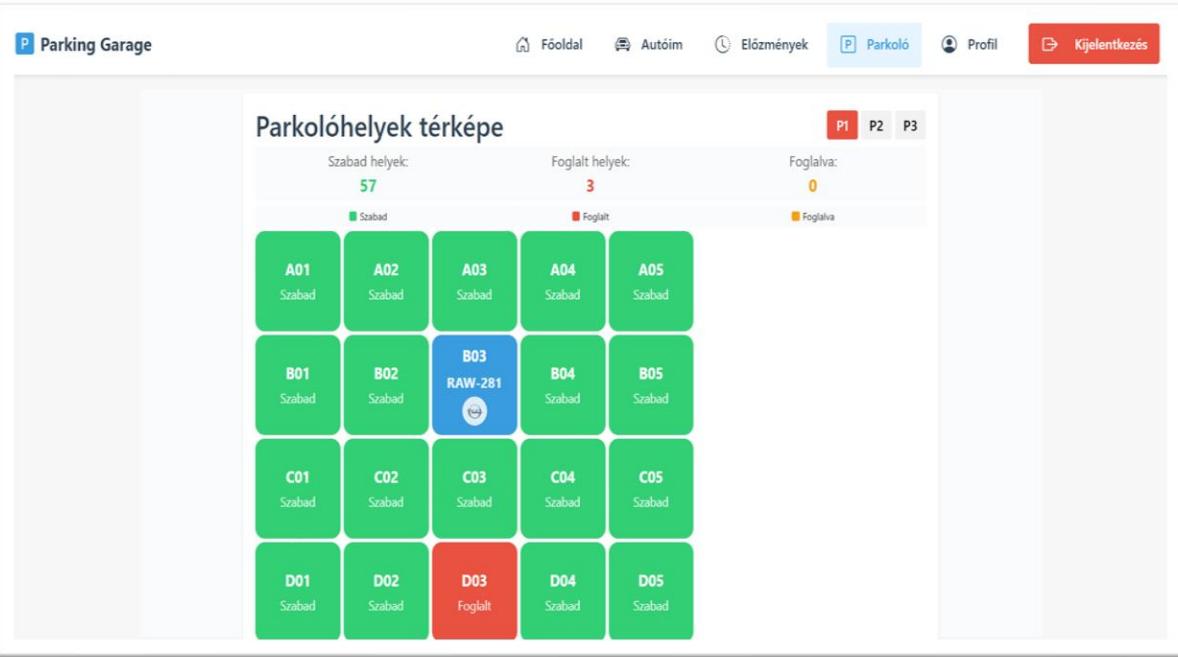
9. ábra: Saját autó adatainak felvétele

### 3. Parkolási Események

- Parkolóhelyek megtekintése és foglalása
- Aktuális parkolási állapot követése
- Parkolási előzmények megtekintése
- Parkolási díjak és számlák kezelése



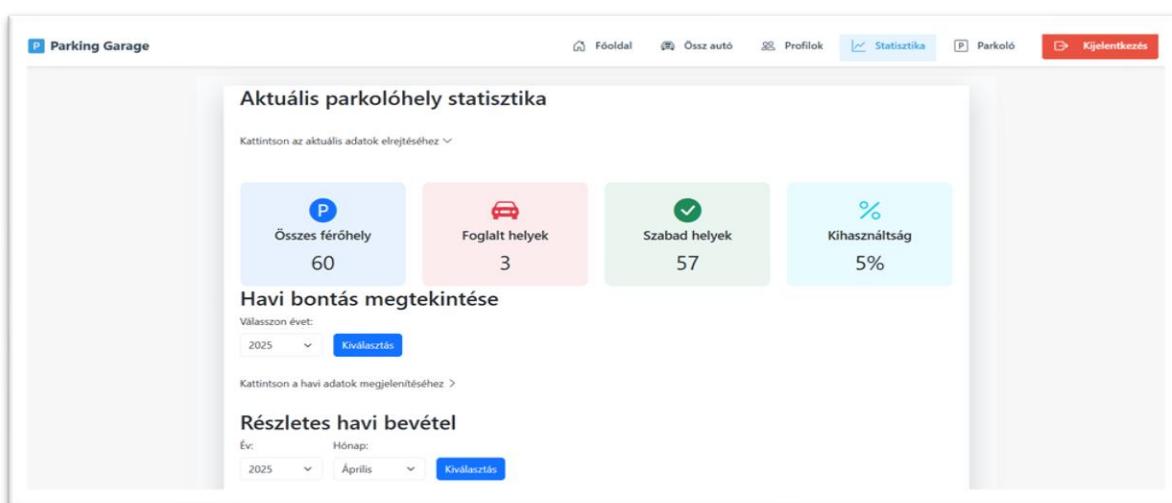
10.ábra: Parkolóhely kiválasztása



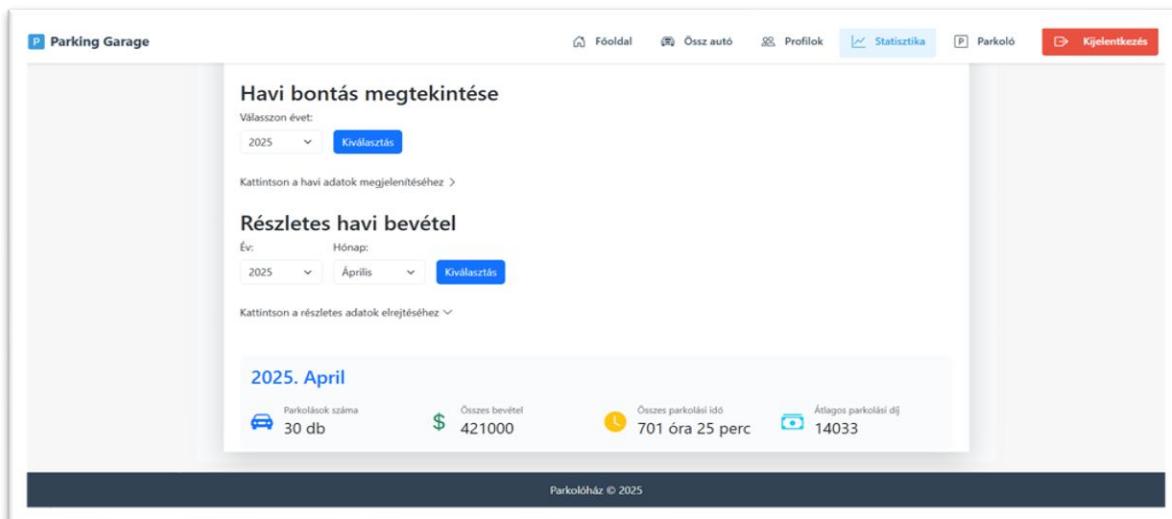
11. ábra: Saját autó megjelenítése a parkoló fül alatt

#### 4. Admin Panel

- Felhasználók kezelése
- Parkolóhelyek üzemeltetése
- Statisztikák és riportok megtekintése
- Számlák kezelése és újraküldése
- Rendszerbeállítások módosítása



12. ábra: Statisztikai adatok – aktuális parkolóhelyek



13. ábra: Statisztikai adatok havi lebontása

**Összes autó**

<b>Toyota Corolla</b> Rendszám: ABC-123 Évjárat: 2020 Tulajdonos: John Doe Email: john.doe@example.com Parkoló állapot: Nincs parkolóban	<b>Mercedes 124</b> Rendszám: HLE-074 Évjárat: 1994 Tulajdonos: John Doe Email: john.doe@example.com Parkoló állapot: Nincs parkolóban	<b>Opel Astra</b> Rendszám: RAW-281 Évjárat: 2011 Tulajdonos: János Scher Email: jani.prog001@gmail.com Parkoló állapot: Parkolóban
<b>Ford Mustang GT</b> Rendszám: FORD-007 Évjárat: 1974 Tulajdonos: János Scher Email: jani.prog001@gmail.com Parkoló állapot: Nincs parkolóban	<b>Audi A4</b> Rendszám: SU-071-JZ Évjárat: 2001 Tulajdonos: Szabolcs Juhász Email: juhaszsabolcs90@gmail.com Parkoló állapot: Nincs parkolóban	<b>BMW X5</b> Rendszám: BOSS12 Évjárat: 2020 Tulajdonos: Szabolcs Juhász Email: juhaszsabolcs90@gmail.com Parkoló állapot: Nincs parkolóban

14. ábra: Összes autó kezelőfelülete

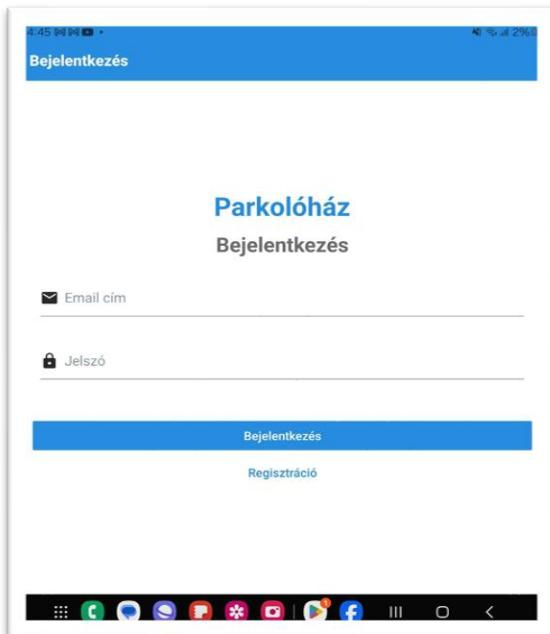
Keresés név alapján...

<b>John Doe</b> Email: john.doe@example.com Telefonszám: 1234567890 Autók száma: 2 Autók megtékinése	<b>Admin User</b> Email: admin@example.com Telefonszám: 0987654321 Autók száma: 0	<b>János Scher</b> Email: jani.prog001@gmail.com Telefonszám: 06306036313 Autók száma: 2 Autók megtékinése
Toyota Corolla 2020 ABC-123 Nincs parkolóban	Mercedes 124 1994 HLE-074 Nincs parkolóban	Opel Astra 2011 RAW-281 Parkolóban
Szabolcs Juhász Email: szabolcs.juhasz@example.com Telefonszám: 091234567890 Autók száma: 0	Juhász-Szabó Jetta Email: juhasz-szabot@szabot.hu Telefonszám: 06543210987 Autók száma: 0	Major Attila Email: attila.major@attila.hu Telefonszám: 06543210987 Autók száma: 0

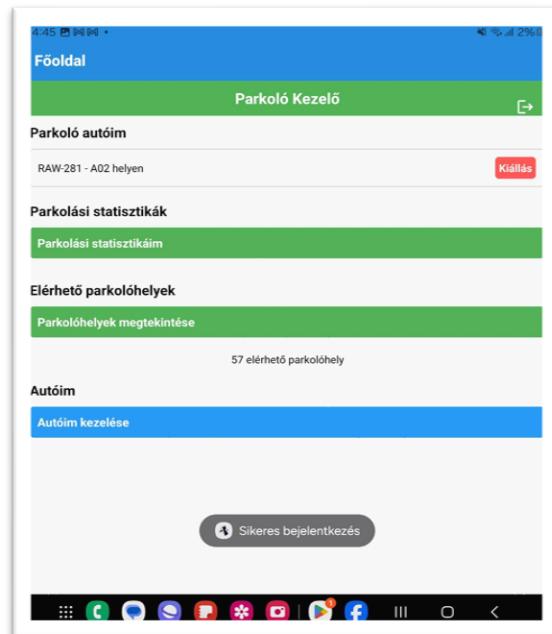
15. ábra: Felhasználók kezelőfelülete

## 5. Mobil applikáció

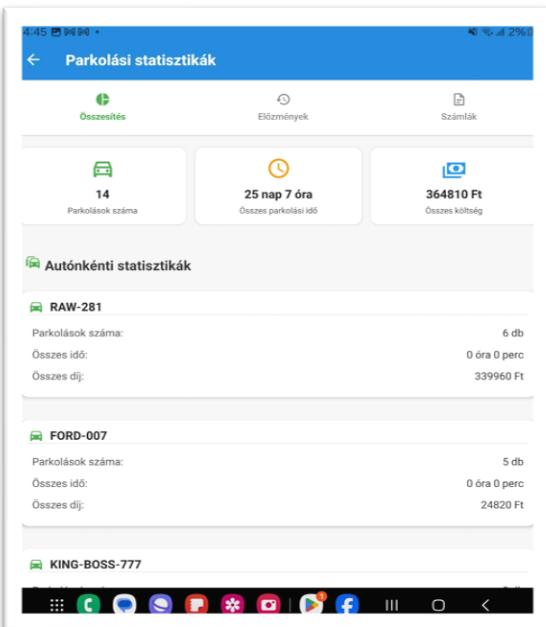
- Egyszerű, átlátható felület
- Könnyű kezelhetőség



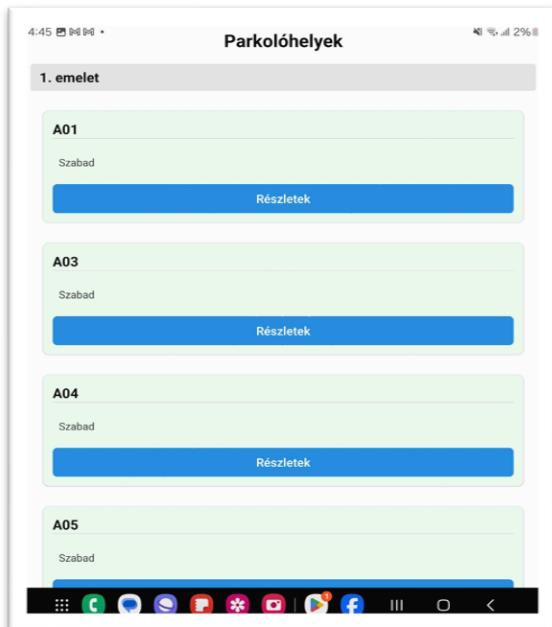
16. ábra: Bejelentkezés



17. ábra: Parkoláskezelő



18. ábra: Parkolási statisztikák



19. ábra: Parkolóhelyek

### **A felület főbb jellemzői:**

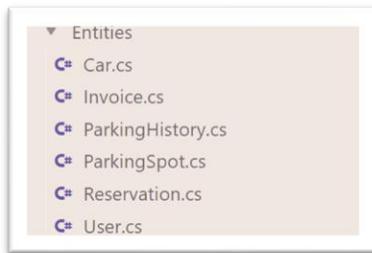
- Modern, reszponzív dizájn
- Intuitív navigáció
- Valós idejű visszajelzések
- Automatikus email értesítések
- Biztonságos adatkezelés
- Felhasználóbarát hibaüzenetek

A frontend a Netlify szolgáltatáson keresztül van üzemeltetve, amely alapvetően megbízható működést biztosít, azonban időnként lassabb betöltést eredményezhet. A rendszer automatikusan frissül az új verziókkal, és a felhasználók mindenkorban a legfrissebb verziót látják.

## Adatbázisok és eszközök részletes bemutatása

---

Az alkalmazás MySQL adatbázist használ, amelyet az Aiven szolgáltatáson keresztül üzemeltetünk. Az adatbázis relációs adatmodellre épül, és az Entity Framework Core 8.0 ORM keretrendszer segítségével kezeljük. Az adatbázisban a következő fő entitások találhatók:



20. ábra: Adatbázis entitások

### 1. User (Felhasználó)

- Azonosító, email, jelszó hash, név, telefonszám, admin jogosultság
- Kapcsolatok: járművek, parkolási események, számlák

### 2. Car (Jármű)

- Azonosító, rendszám, márka, modell, évjárat, parkolási státusz
- Kapcsolatok: felhasználó, parkolóhely

### 3. ParkingSpot (Parkolóhely)

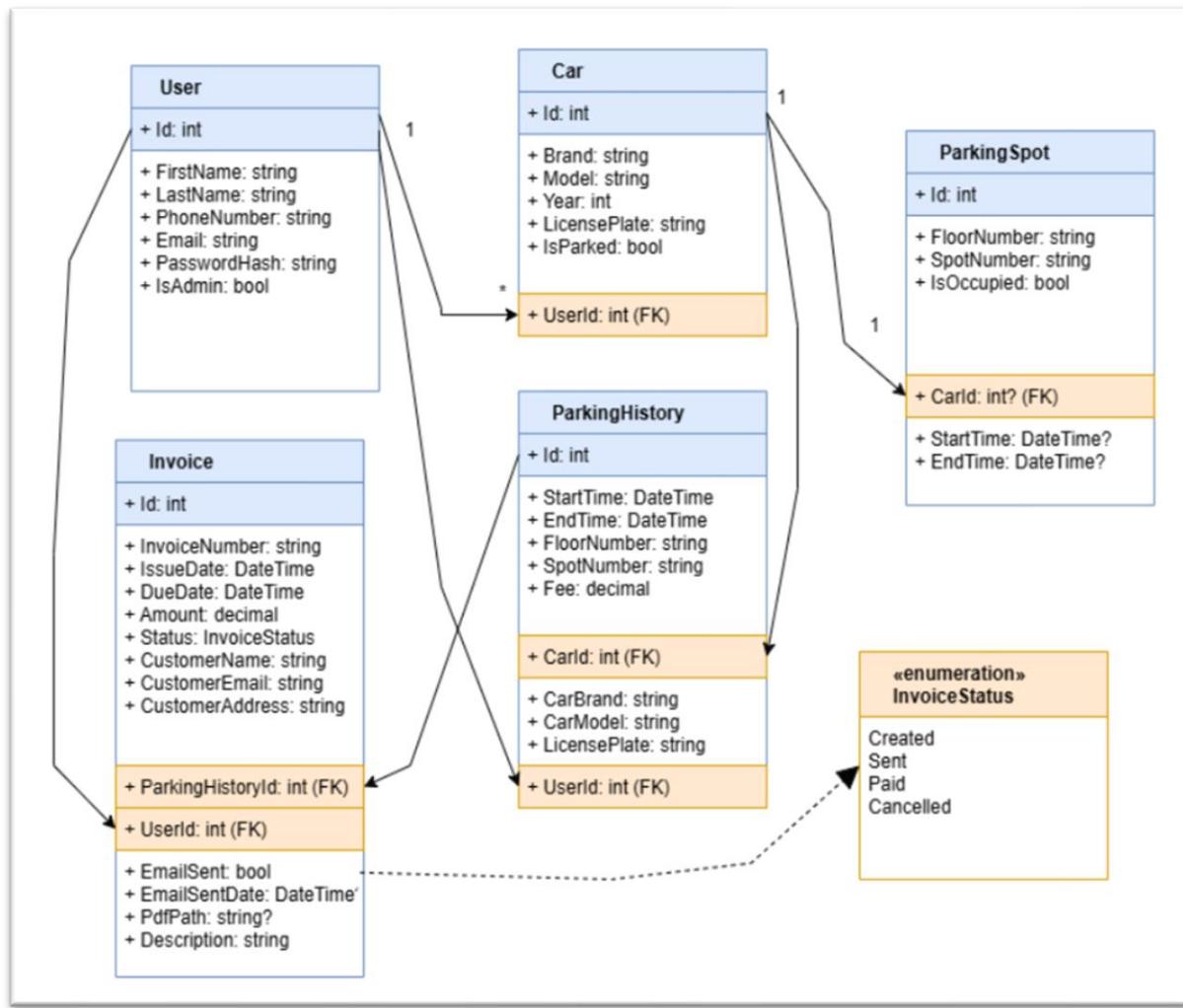
- Azonosító, emelet szám, helyszám, foglaltság
- Kapcsolatok: jármű, parkolási események

### 4. ParkingHistory (Parkolási Előzmény)

- Azonosító, kezdő időpont, záró időpont, díj
- Kapcsolatok: felhasználó, jármű, parkolóhely

## 5. Invoice (Számla)

- Azonosító, számlaszám, összeg, kiállítás dátuma, esedékesség
- Kapcsolatok: felhasználó, parkolási esemény



21.ábra: Entitás-kapcsolat (ER) diagram

# Fejlesztői Környezet

---

A projekt fejlesztéséhez a következő eszközöket és technológiákat használtuk:

## 1. Backend

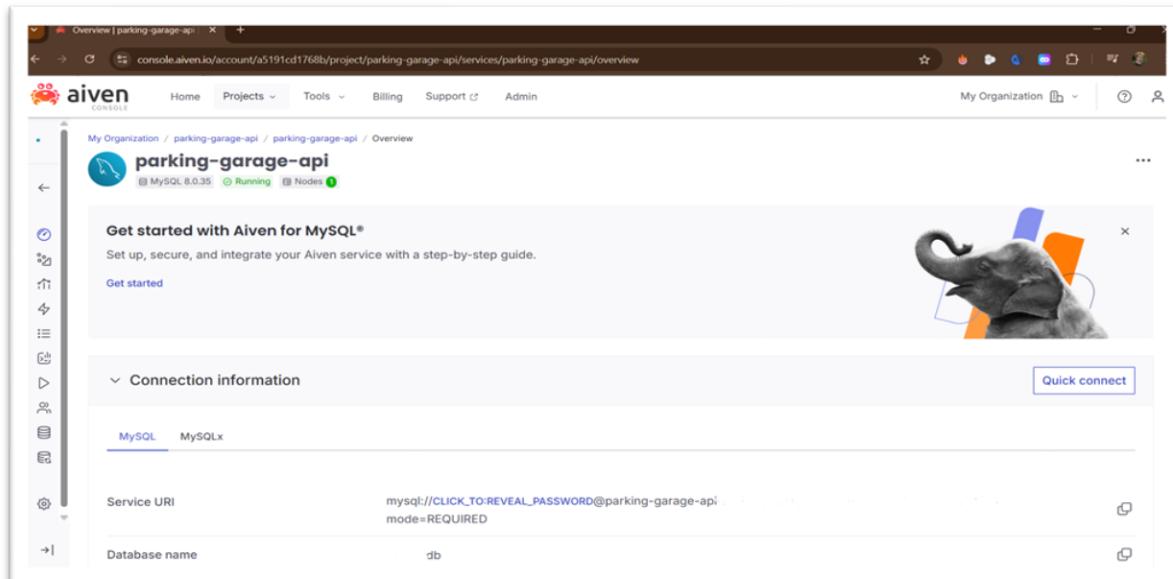
- ASP.NET Core 8.0
- Entity Framework Core 8.0
- Pomelo.EntityFrameworkCore.MySql
- Swagger/OpenAPI
- MailKit (email küldés)
- iTextSharp (PDF generálás)

## 2. Frontend

- Svelte

## 3. Adatbázis

- MySQL 8.0
- Aiven üzemeltetés



22. ábra: Aiven felület – MySQL adatbázis kapcsolati adatok

## 4. Deployment

- Render (backend)

The screenshot shows the Render.com dashboard for the 'ParkingGarageApiBackend' service. The left sidebar contains navigation links for Environment, Events, Settings (which is selected), MONITOR (Logs, Metrics), MANAGE (Environment, Shell, Scaling, Previews, Disks, Jobs), Changelog, Invite a friend, Contact support, and Render Status. The main panel displays the 'WEB SERVICE' section for 'ParkingGarageApiBackend'. It shows the Docker icon, a 'Free' plan, and a link to upgrade the instance. The URL is https://parkinggarageapibackend.onrender.com. A 'Connect' button and a 'Manual Deploy' button are present. Below this, the 'Settings' section has a 'General' tab. Under 'Name', it says 'ParkingGarageApiBackend' with an 'Edit' link. Under 'Region', it says 'Frankfurt (EU Central)'. Under 'Instance Type', it shows 'Free' with 0.1 CPU and 512 MB, with an 'Update' button. A note says 'Please ask the owner (juhaszszabolcs90@gmail.com) to enter their payment information.'

23. ábra: Backend hosztolás Render.com felületen

The screenshot shows the Render.com dashboard for the 'ParkingGarageApiBackend' service, focusing on the deployment log. The left sidebar is identical to the previous screenshot. The main panel shows a list of deployment events:

- Deploy live for 624a17a: Merge pull request #6 from janiprog001/admin-autotorles User statisztikák listázásának javítása (April 7, 2025 at 10:20 PM)
- Deploy started for 624a17a: Merge pull request #6 from janiprog001/admin-autotorles User statisztikák listázásának javítása (Manually triggered by you via Dashboard, April 7, 2025 at 10:19 PM)
- Deploy live for 8ff1916: Merge pull request #5 from janiprog001/admin-autotorles Admin autotorles (April 7, 2025 at 5:35 PM)
- Deploy started for 8ff1916: Merge pull request #5 from janiprog001/admin-autotorles Admin autotorles (Manually triggered by you via Dashboard, April 7, 2025 at 5:33 PM)
- Deploy live for dc0d651: Merge pull request #2 from janiprog001/parkolas\_statusz Parkolás állapotának lekérésre (April 7, 2025 at 5:31 PM)

Each event includes a 'Rollback' button.

24. ábra: Backend szolgáltatás deploy naplója (Render.com)

- Netlify (frontend)

Sites / parking-garage-app

**Site overview**

**parking-garage-app**

**parking-garage-app.netlify.app**

Deploys from GitHub.  
Published on Apr 13.

**Site configuration** **Favorite site** **Enable visual editor**

**Set up your site**

- Your site is deployed ✓  
Try a test build and deploy, directly from your Git repository or a folder.
- Set up a custom domain  
Buy a [new domain](#) → or set up a [domain you already own](#) →
- Secure your site with HTTPS  
Your site is secured automatically with a Let's Encrypt certificate.

25. ábra: Frontend alkalmazás kezelése a Netlify-on

Sites / parking-garage-app

**Deploys**

**Deploys for parking-garage-app**

**parking-garage-app.netlify.app**

Deploys from [github.com/juhasz-szabolcs/Parking\\_Garage\\_Frontend](#).  
Published [main@026c8a4](#)

Auto publishing is on. Deploys from [main](#) are published automatically.

**Deploy settings** **Notifications** **Lock to stop auto publishing**

**Test your site's Lighthouse performance**

Want to see how your site will perform before you deploy? Install the Lighthouse plugin for build-time Lighthouse scores and reports.

[Learn more](#)

[Install Lighthouse plugin](#)

**Search by branch name or deploy ID**

Filter: Any status Any time frame

Deployment	Date	Time
Production: main@026c8a4 <a href="#">Published</a>	Apr 13 at 7:51 PM	Deployed in 32s
Merge pull request #15 from juhasz-szabolcs/car_id		
Deploy Preview #15: car_id@6aaaf0a7	Apr 13 at 7:51 PM	Deployed in 37s
Ft jelölése		
Production: main@c2f7b15	Apr 12 at 1:01 PM	Deployed in 34s
Merge pull request #14 from juhasz-szabolcs/car_id		

26. ábra: Frontend deploy napló a Netlify felületén

## 5. Verziókezelés

- Github

The screenshot shows the GitHub repository page for 'janiprog001/ParkingGarageAPI'. The main branch is 'main', which has 73 commits. The commits are listed in a table with columns for file name, description, and date. Key commits include: 'Admin statisztikai funkciók implementálása' (last month), 'Változtatások a mobilapp működéséhez' (yesterday), 'Merge branch 'main' of https://github.com/janiprog001/ParkingGarageAPI' (yesterday), 'LoginDto létrehozása' (last month), 'Változtatások a mobilapp működéséhez' (yesterday), 'InMemory DB-ról MySQL-re átállás' (last month), 'InMemory DB-ról MySQL-re átállás' (last month), 'InMemory DB-ról MySQL-re átállás' (last month), 'Számlázási rendszer implementálása' (last month), 'Docker létrehozása' (last month), 'Adatbázis migrálása a Aiven felületre.' (last month), and 'Hibajavítás a Render-en való futtatáshoz' (last month). The repository has 1 watch, 0 forks, and 0 stars.

27. ábra: Backend verziókezelése a GitHub felületén

The screenshot shows the GitHub repository page for 'juhasz-szabolcs/Parking\_Garage\_Frontend'. The main branch is 'main', which has 126 commits. The commits are listed in a table with columns for file name, description, and date. Key commits include: 'Cars fájl formázása' (last month), 'Ft jelölése' (2 weeks ago), 'Parkoló logó lecerélése' (3 weeks ago), 'fix regisztráció' (last month), 'Init project' (last month), 'macOS és iOS konfigurációs beállítások' (3 weeks ago), 'redirect javítása' (last month), 'netlify adapter instalálása' (last month), 'netlify adapter instalálása' (last month), and 'netlify adapter instalálása' (last month). The repository has 1 watch, 0 forks, and 0 stars. It also includes a 'Languages' section showing Svelte (89.4%), JavaScript (10.2%), and HTML (0.4%).

28. ábra: Frontend verziókezelése a GitHub felületén

A rendszer moduláris felépítésű, ami lehetővé teszi a könnyű bővíthetőséget és karbantarthatóságot. Az adatbázis struktúra rugalmas, az API jól dokumentált, és a kód architektúra tiszta, ami megkönnyíti a későbbi fejlesztéseket.

# Minőségbiztosítás és tesztelési folyamatok

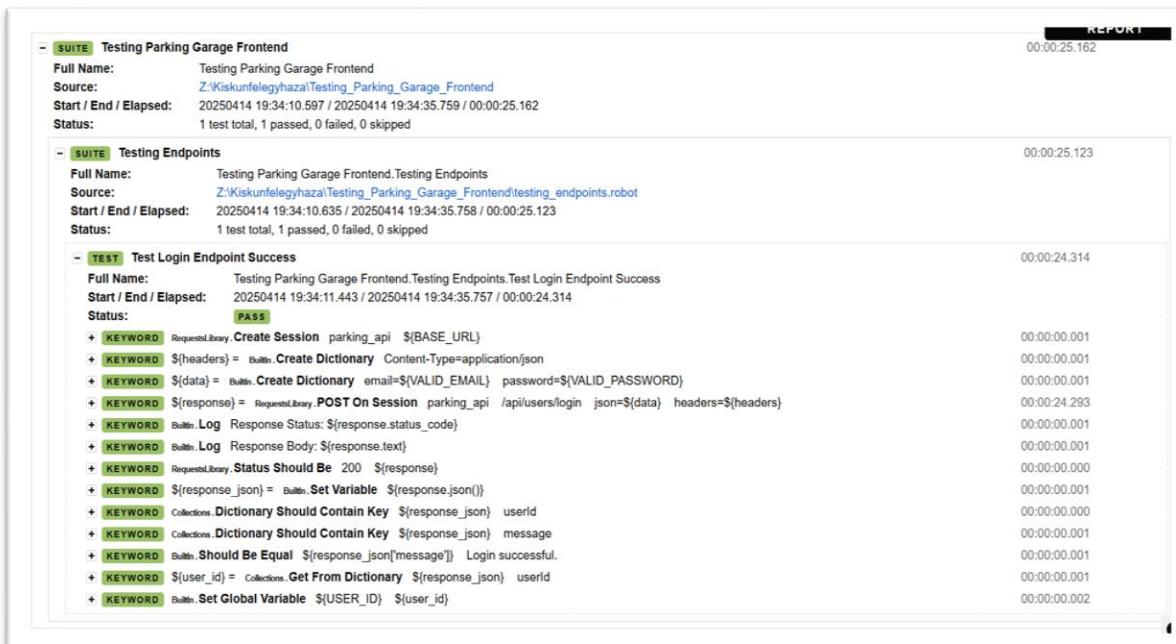
## API Tesztelés

### 1. Swagger Integráció

- A rendszer Swagger/OpenAPI dokumentációt használ
- A Swagger UI elérhető a fejlesztési környezetben: <http://localhost:5025/swagger>
- Az API végpontok dokumentációja és tesztelési felülete automatikusan generálódik

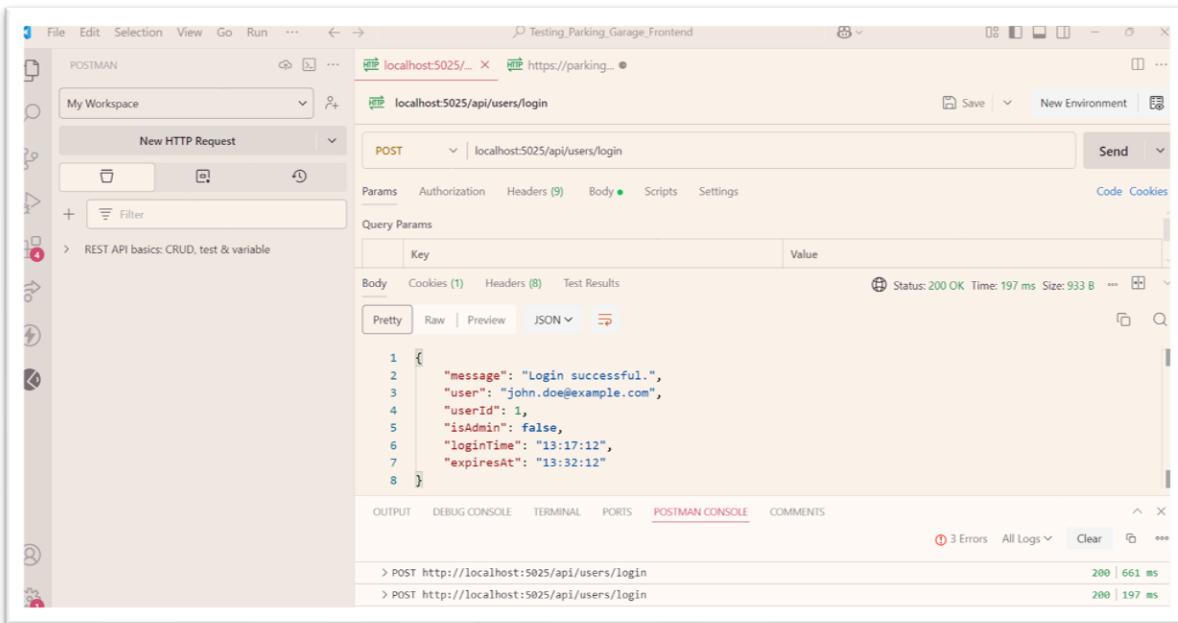
### 1. API Végpont Tesztelés

- Robot Framework végpont tesztelés:



29. ábra: Robot Framework – API-végpont tesztelési riport

- Postman tesztelés



30. ábra: Felhasználói bejelentkezés tesztelése Postman segítségével

- A végpontok tesztelése a Swagger UI-on keresztül is történik
- A válaszok és hibakezelések ellenőrzése

The screenshot shows the AdminStatistics endpoint group in the ParkingGarageAPI v1.0 OAS3 Swagger UI. It contains five GET requests:

- GET /api/admin/statistics/all-history
- GET /api/admin/statistics/revenue
- GET /api/admin/statistics/occupancy
- GET /api/admin/statistics/user-activity
- GET /api/admin/statistics/monthly-revenue

31. ábra: AdminStatistics végpontok a Swagger UI-ban

The screenshot shows the Car (autó) entitás API-végpontok a Swagger UI-ban. It contains four operations:

- GET /api/cars
- POST /api/cars
- DELETE /api/cars/{id}
- GET /api/cars/all

32. ábra: Car (autó) entitás API-végpontok a Swagger UI-ban

### Invoice

- GET /api/invoices
- GET /api/invoices/{id}
- GET /api/invoices/{id}/download
- POST /api/invoices/{id}/resend
- PUT /api/invoices/{id}/status

33. ábra: Invoice (számla) végpontok a Swagger UI-ban

### Parking

- GET /api/parking/spots/available
- GET /api/parking/spots
- POST /api/parking/start
- POST /api/parking/end
- GET /api/parking/my
- GET /api/parking/status/{carId}

34. ábra: Parking (parkolás) végpontok a Swagger UI-ban

### Statistics

- GET /api/statistics/history
- GET /api/statistics/summary
- GET /api/statistics/by-car
- GET /api/statistics/monthly

35. ábra: Statistics (statisztika) végpontok a Swagger UI-ban

### Test

- GET /api/test/userdata

36. ábra: Test végpont a Swager UI-ban

### Users

- POST /api/users/register
- GET /api/users
- GET /api/users/{id}
- PUT /api/users/{id}
- DELETE /api/users/{id}
- POST /api/users/login
- POST /api/users/logout

37. ábra: User (felhasználó) végpontok a Swager UI-ban

## Frontend Tesztelés

### 1. API Integráció Tesztelés

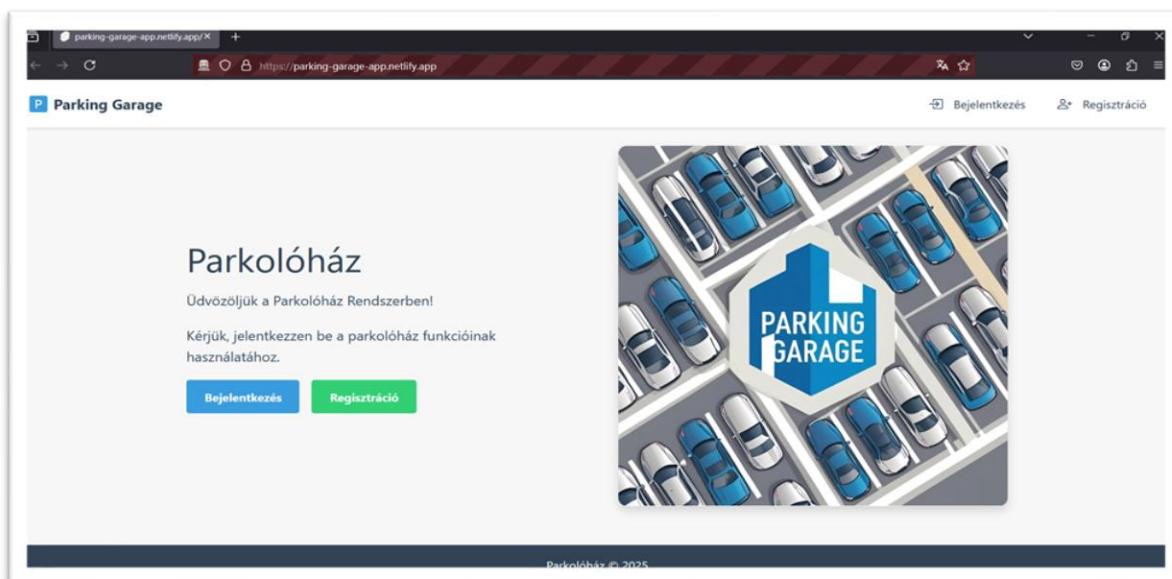
- Az api.js fájlban részletes hibakezelés és naplózás implementálva
- A kérések és válaszok részletes naplózása fejlesztési célokra
- CORS beállítások tesztelése különböző domainek között

### 2. Reszponzív Tesztelés

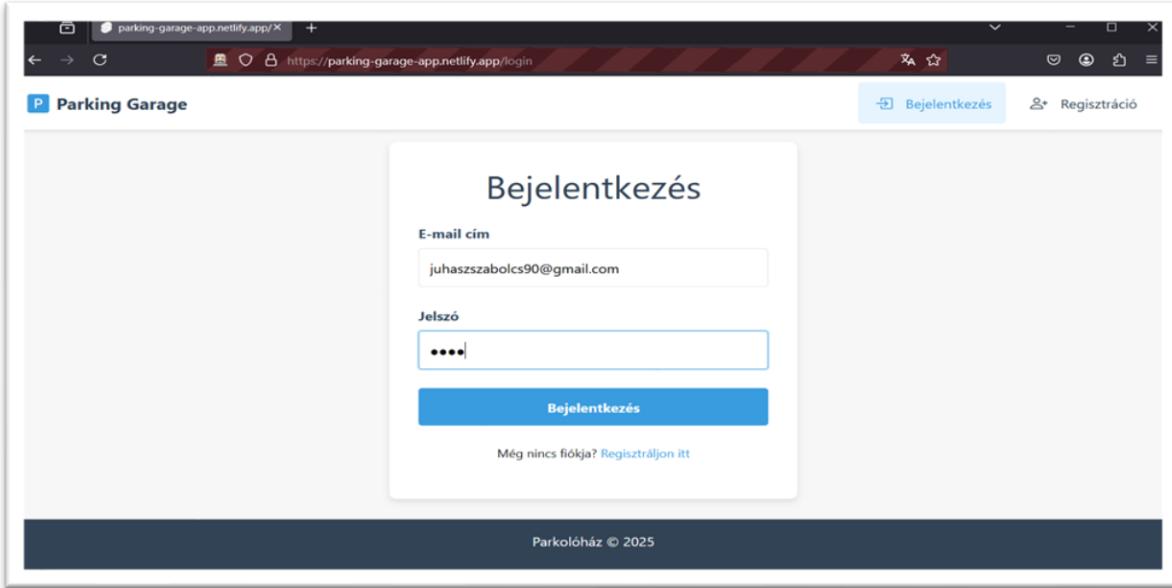
- A frontend reszponzív dizájn tesztelése
- CORS beállítások konfigurálva a különböző környezetekhez:
- Lokális fejlesztés: http://localhost:5173
- Netlify deployment: https://parking-garage-app.netlify.app
- Render deployment: https://\*.onrender.com

### 3. Automata tesztelés

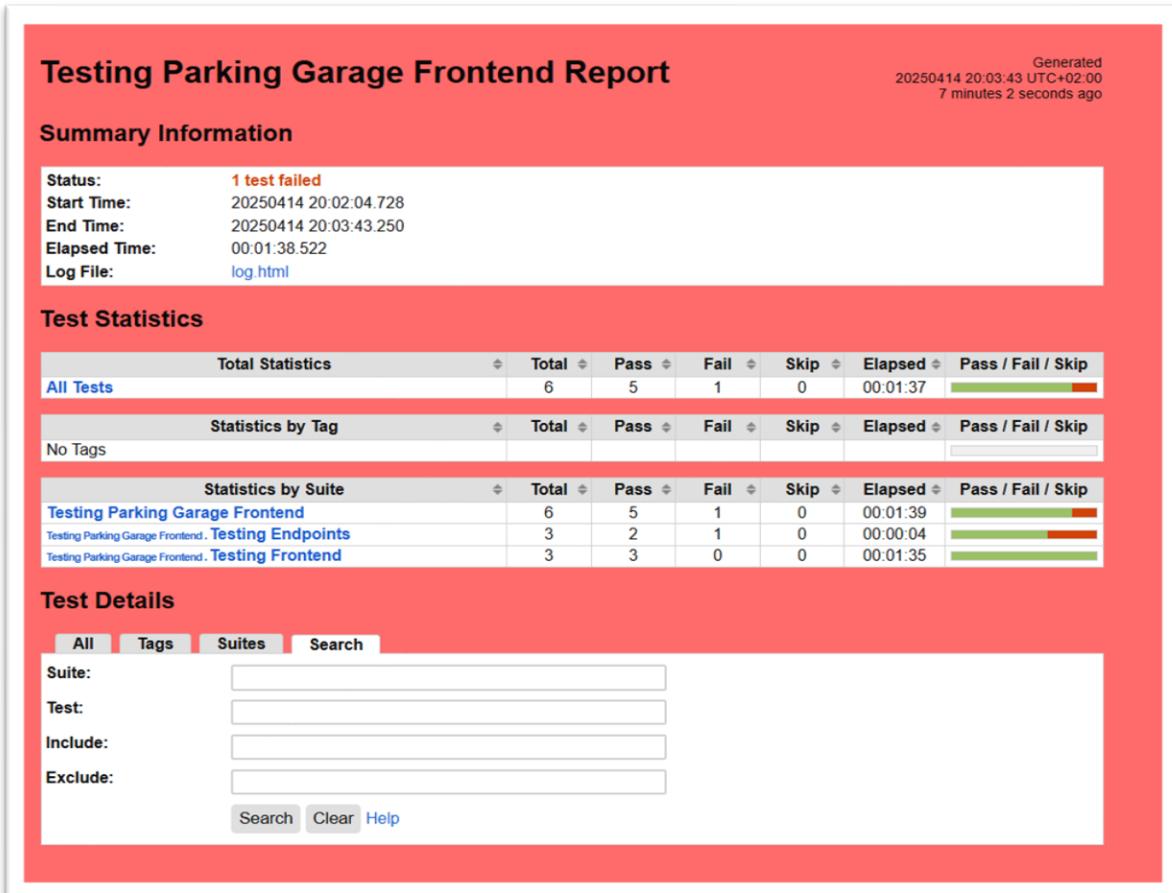
1. Teszeset létrehozása a bejelentkezésre Robot Framework-el:



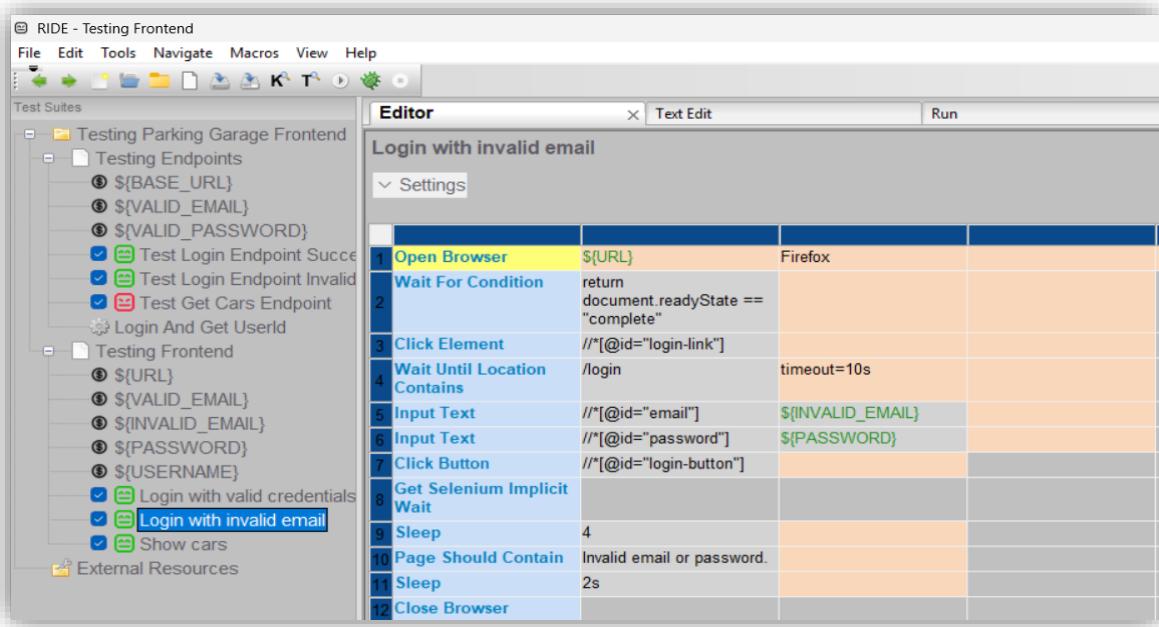
38. ábra: Frontend automatizált tesztelés Robot Framework segítségével



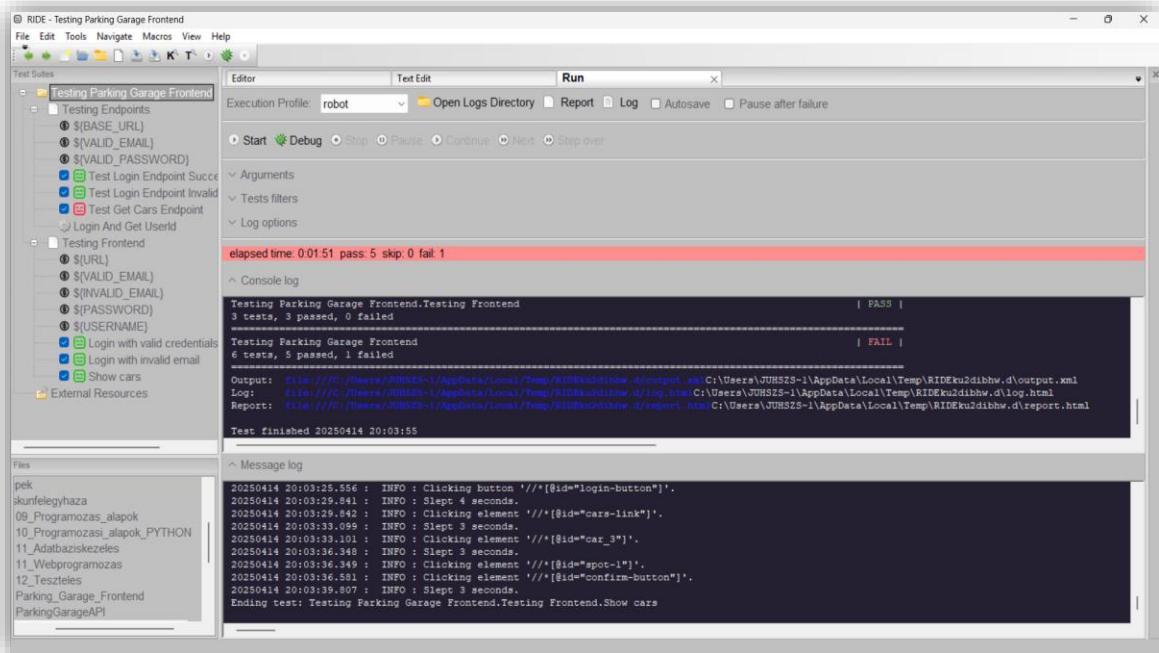
39. ábra: Bejelentkezés automatizált tesztelés Robot Framework segítségével



40. ábra: Frontend tesztek összesített eredménye (Robot Framework)



41. ábra: Robot Framework RIDE: Érvénytelen e-maillel történő bejelentkezés teszeset



42. ábra: Robot Framework RIDE: Frontend tesztek futtatása és eredményei

## Környezeti Tesztelés

### 1. Fejlesztői Környezet

- Lokális fejlesztés: <http://localhost:5025>
- Adatbázis seed tesztelés
- Környezeti változók kezelése

### 2. Produkciós Környezet

- Render deployment tesztelés
- Netlify deployment tesztelés
- Aiven MySQL kapcsolat tesztelése

## Biztonsági Tesztelés

### 1. Autentikáció

- Cookie alapú hitelesítés tesztelése
- Jogosultságkezelés ellenőrzése
- Admin felület védelme

### 2. Adatbázis Biztonság

- SSL kapcsolat tesztelése
- Környezeti változók kezelése
- Adatbázis kapcsolat timeout kezelése

A tesztelés főként a Swagger UI-on keresztül és manuális ellenőrzésekkel történik. Néhány automata tesztet is végeztünk, de nem ez volt a jellemző. A frontend oldalon az API hívások részletes naplózása segíti a hibakeresést és tesztelést.

A rendszer reszponzivitását különböző eszközökön is kipróbáltuk (asztali gép, tablet, mobiltelefon). A tesztelésre megkért felhasználóktól kapott visszajelzések alapján folyamatosan javítottuk az élményt és hibákat.

## Összegzés

---

A **Parkolóház Kezelő Rendszer** projekt során egy olyan alkalmazást hoztunk létre, amely **valóban hasznosítható egy valós parkolóház működtetéséhez**. A rendszer fő erőssége, hogy **egyszerűen használható**, miközben minden szükséges funkciót tartalmaz egy modern parkolóház üzemeltetéséhez.

A projekt során sikeresen implementáltuk a moduláris architektúrát, amely lehetővé teszi a rendszer jövőbeli bővítését és karbantartását. A fejlesztés során alkalmazott modern technológiák (ASP.NET Core 8.0, Svelte, MySQL) és fejlesztési módszerek (Git verziókezelés, Trello projektmenedzsment) biztosítják a rendszer hosszú távú fenntarthatóságát. A kiterjedt tesztelési folyamatok és a minőségbiztosítási intézkedések garantálják a rendszer megbízható működését mind a felhasználók, mind az üzemeltetők számára.

A projekt készítése során rengeteget tanultunk, hiszen ez volt az első igazi nagy projektünk. A csapatmunkától kezdve a modern fejlesztési technológiák használatán át a projektmenedzsmentig szinte minden területen tapasztalatot szereztünk. A már meglévő készségeinket, mint a hatékony kommunikáció és együttműködés, a verziókezelés és dokumentáció készítés, valamint a komplex rendszerek tervezése és implementálása, jelentősen kibővítettük és gyakorlati tapasztalattal gazdagítottuk. Ezek a tapasztalatok és megszerzett ismeretek nélkülözhetetlen alapot jelentenek a jövőbeli szakmai fejlődésünkhöz.

## Felhasznált források

---

<https://learn.microsoft.com/en-us/dotnet/>

<https://svelte.dev/>

<https://getbootstrap.com/>

<https://github.com/>

<https://www.mysql.com/>

<https://trello.com/>

<https://chat.openai.com/>

<https://robotframework.org/>

<https://robotframework.org/Selenium2Library/Selenium2Library.html>

# Ábrajegyzék

---

1.ábra: Entity Framework Core és MySQL adatbáziskapcsolat konfigurációja.....	5
2. ábra: Felhasználói foglalások lekérdezése a ReservationController-ben.....	6
3. ábra: MySQL kapcsolat környezeti változókkal.....	7
4. ábra: Jelszavak Base64-es kódolása.....	8
5. ábra: SWOT analízis.....	9
6. ábra: Trello.....	11
7. ábra: Főoldal.....	15
8. ábra: Saját autók megjelenítése.....	16
9. ábra: Saját autó adatainak felvétele.....	16
10.ábra: Parkolóhely kiválasztása.....	17
11. ábra: Saját autó megjelenítése a parkoló fül alatt.....	18
12. ábra: Statisztikai adatok – aktuális parkolóhelyek.....	19
13. ábra: Statisztikai adatok havi lebontása.....	19
14. ábra: Összes autó kezelőfelülete.....	20
15. ábra: Felhasználók kezelőfelülete.....	20
16. ábra: Bejelentkezés.....	21
17. ábra: Parkoláskezelő.....	21
18. ábra: Parkolási statisztikák.....	21
19. ábra: Parkolóhelyek.....	21
20. ábra: Adatbázis entitások.....	23
21.ábra: Entitás-kapcsolat (ER) diagram.....	24
22. ábra: Aiven felület – MySQL adatbázis kapcsolati adatok.....	25
23. ábra: Backend hosztolás Render.com felületen.....	26
24. ábra: Backend szolgáltatás deploy naplója (Render.com).....	26
25. ábra: Frontend alkalmazás kezelése a Netlify-on.....	27
26. ábra: Frontend deploy napló a Netlify felületén.....	27

27. ábra: Backend verziókezelése a GitHub felületén.....	28
28. ábra: Frontend verziókezelése a GitHub felületén.....	28
29. ábra: Robot Framework – API-végpont tesztelési riport.....	29
30. ábra: Felhasználói bejelentkezés tesztelése Postman segítségével.....	30
31. ábra: AdminStatistics végpontok a Swagger UI-ban.....	31
32. ábra: Car (autó) entitás API-végpontok a Swagger UI-ban.....	31
33. ábra: Invoice (számla) végpontok a Swagger UI-ban.....	32
34. ábra: Parking (parkolás) végpontok a Swagger UI-ban.....	32
35. ábra: Statistics (statisztika) végpontok a Swagger UI-ban.....	33
36. ábra: Test végpont a Swager UI-ban.....	33
37. ábra: User (felhasználó) végpontok a Swagger UI-ban.....	33
38. ábra: Frontend automatizált tesztelés Robot Framework segítségével.....	34
39. ábra: Bejelentkezés automatizált tesztelés Robot Framework segítségével.....	35
40. ábra: Frontend tesztek összesített eredménye (Robot Framework).....	35
41. ábra: Robot Framework RIDE: Érvénytelen e-maillel történő bejelentkezés teszteset.....	36
42. ábra: Robot Framework RIDE: Frontend tesztek futtatása és eredményei.....	36