

F

Pannonhalmi Főapátság Szegedi SOB Technikuma



Parkolóház Kezelő Rendszer

Szoftverfejlesztő és -tesztelő

Készítették: Juhász Szabolcs, Scher János, Major Attila

Témavezető: Rédei Dávid

2025

Szeged

Tartalom

A projekt áttekintése	3
A projekt céljának részletes ismertetése	3
Tervezési szempontok és alkalmazott módszerek	4
SWOT elemzés	7
Feladatmegosztás a projektcsapaton belül	8
Ütemterv – a projekt előrehaladásának menete	10
Felhasználói felület és alkalmazott technológiák	13
Adatbázisok és eszközök részletes bemutatása.....	21
Minőségbiztosítás és tesztelési folyamatok	29
Összegzés.....	36
Felhasznált források	38

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 projekt céljának részletes ismertetése

A fejlesztés célja egy **komplex, modulárisan felépített parkolóház-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 parkolóházak 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 fázist. 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:

1. Entity Framework Core konfiguráció (Program.cs):

 csharp

```
builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseMySQL(connectionString,
        new MySqlServerVersion(new Version(8, 0, 13)),
        mysqlOptions => mysqlOptions.EnableRetryOnFailure()));
```

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

Példa a projektből:

```
csharp

try
{
    // Adatbázis művelet
    var user = await context.Users.FirstOrDefaultAsync(u => u.Email == email);
    if (user == null)
    {
        return NotFound("Felhasználó nem található.");
    }

    // Üzleti logika
    if (!user.IsAdmin)
    {
        return Forbid("Nincs jogosultsága.");
    }

    // További műveletek...
}
catch (DbUpdateException ex)
{
    // Adatbázis hibák kezelése
    return StatusCode(500, "Adatbázis hiba történt.");
}
catch (Exception ex)
{
    // Általános hibák kezelése
    return StatusCode(500, "Váratlan hiba történt.");
}
```

A frontend technológiai alapját 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:

1. Program.cs - Adatbázis kapcsolat konfigurálása:

```
csharp

// MySQL kapcsolat környezeti változókból
var host = Environment.GetEnvironmentVariable("MYSQL_HOST");
var port = Environment.GetEnvironmentVariable("MYSQL_PORT");
var database = Environment.GetEnvironmentVariable("MYSQL_DATABASE");
var user = Environment.GetEnvironmentVariable("MYSQL_USER");
var password = Environment.GetEnvironmentVariable("MYSQL_PASSWORD");
var sslMode = Environment.GetEnvironmentVariable("MYSQL_SSL_MODE") ?? "REQUIRED";

// Kapcsolati string összeállítása
var connectionString = $"Server={host};" +
    $"Port={port};" +
    $"Database={database};" +
    $"User={user};" +
    $"Password={password};" +
    $"SslMode={sslMode}";

// DbContext konfigurálása
builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseMySQL(connectionString,
        new MySqlServerVersion(new Version(8, 0, 13)),
        mySqlOptions => mySqlOptions.EnableRetryOnFailure() // Automatikus újracatlakozás hiba esetén
    ));
```

A jelenlegi megvalósításban a felhasználói 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 **kriptográfiailag biztonságos, egyirányú hash-elési algoritmus** – például a **bcrypt** – bevezetése, amely jelentősen növelné a rendszer adatbiztonságát, különösen a jelszavak kezelése terén:

1. Regisztráció során (UsersController.cs):

```
csharp

user.PasswordHash = Convert.ToBase64String(Encoding.UTF8.GetBytes(user.PasswordHash));
```

A fejlesztés során alkalmazott tervezési alapelvek:

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

Feladatmegosztás a projektcsoporton 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

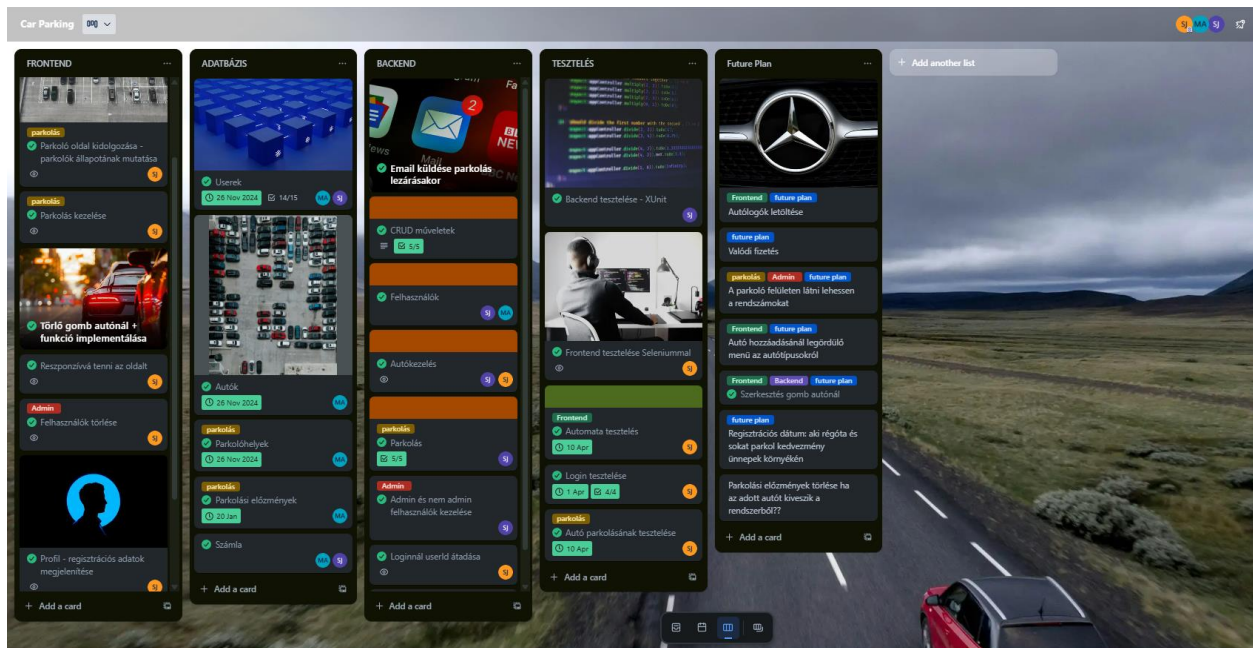
dolgozott a mobilalkalmazás fejlesztésén is, hogy a felhasználók okostelefonon is kényelmesen használhassák a parkolóház rendszert.

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:



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, státuszuk kezelését és az előzmények megtekintését.

10. Hét: Statisztikák és Ríjportok

Kialakítottuk a parkolási előzmények, összesítők, autónkénti 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.

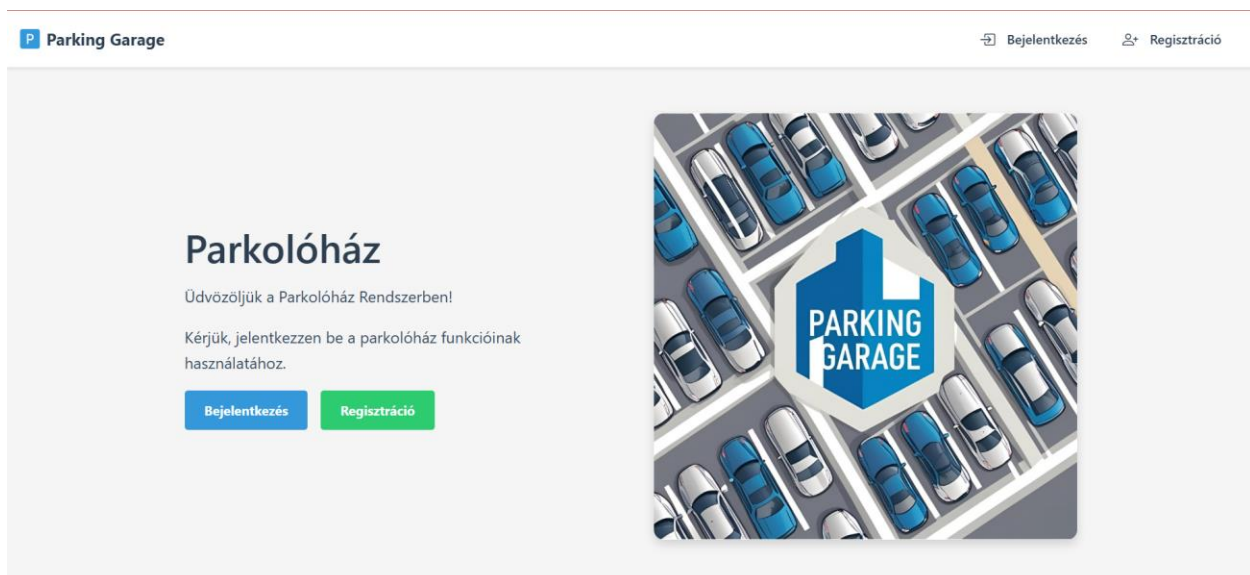
A projekt során a heti ütemezés szerint haladtunk, és minden modul fejlesztése összhangban történt az előre megtervezett lépésekkel. A backend ASP.NET Core 8 technológiával, a frontend Svelte alapokon készült. A fejlesztés során külön figyelmet fordítottunk a reszponzív felhasználói felületre, valamint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



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

Opel Astra	Ford Mustang GT
Rendszám: RAW-281	Rendszám: FORD-007
Évjárat: 2011	Évjárat: 1974
Jelenleg parkol	Nincs parkolva
Parkolás leállítása Törlés	Parkolás indítása Törlés

 Főoldal

 Autóim

 Előzmények

Új autó hozzáadása

Márka*

pl. Toyota

Modell*

pl. Corolla

Évjárat

2025

Rendszám*

pl. ABC-123

Mégse

Mentés

Parkolóház © 2025

3. Parkolási Események

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

Parkolóhely kiválasztása



Válassz parkolóhelyet

P1

P2

P3

Szabad helyek:

57

Foglalt helyek:


3

Foglalva:

0

 Szabad

 Foglalt

 Kiválasztva

A01 Kiválasztva	A02 Szabad	A03 Szabad	A04 Szabad	A05 Szabad
B01 Szabad	B02 Szabad	B03 Foglalt	B04 Szabad	B05 Szabad
C01 Szabad	C02 Szabad	C03 Szabad	C04 Szabad	C05 Szabad
D01 Szabad	D02 Szabad	D03 Foglalt	D04 Szabad	D05 Szabad

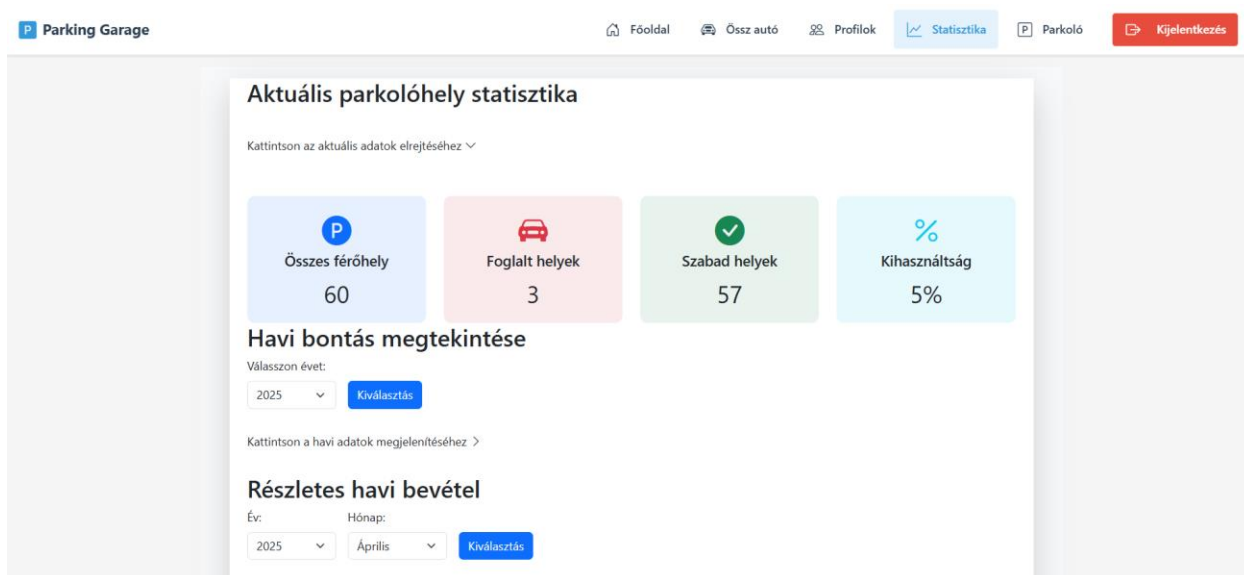
Kiválasztott parkolóhely: A01

Parkolás indítása



1. Admin Panel

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



Parking Garage

Főoldal

Össz autó

Profilok

Statistika

Parkoló

Kijelentkezés

Aktuális parkolóhely statisztika

Kattintson az aktuális adatok megjelenítéséhez >

Havi bontás megtekintése

Válasszon évet:

2025

Kiválasztás

Kattintson a havi adatok elrejtéséhez v

2025

January

0 parkolás

\$ 0 Ft

February

0 parkolás

\$ 0 Ft

March

78 parkolás

\$ 20560 Ft

Parking Garage

Főoldal

Össz autó

Profilok

Statistika

Parkoló

Kijelentkezés

Havi bontás megtekintése

Válasszon évet:

2025

Kiválasztás

Kattintson a havi adatok megjelenítéséhez >

Részletes havi bevétel

Év:

Hónap:

2025

Április

Kiválasztás

Kattintson a részletes adatok elrejtéséhez v

2025. April

Parkolások száma

30 db

Összes bevétel

\$ 421000

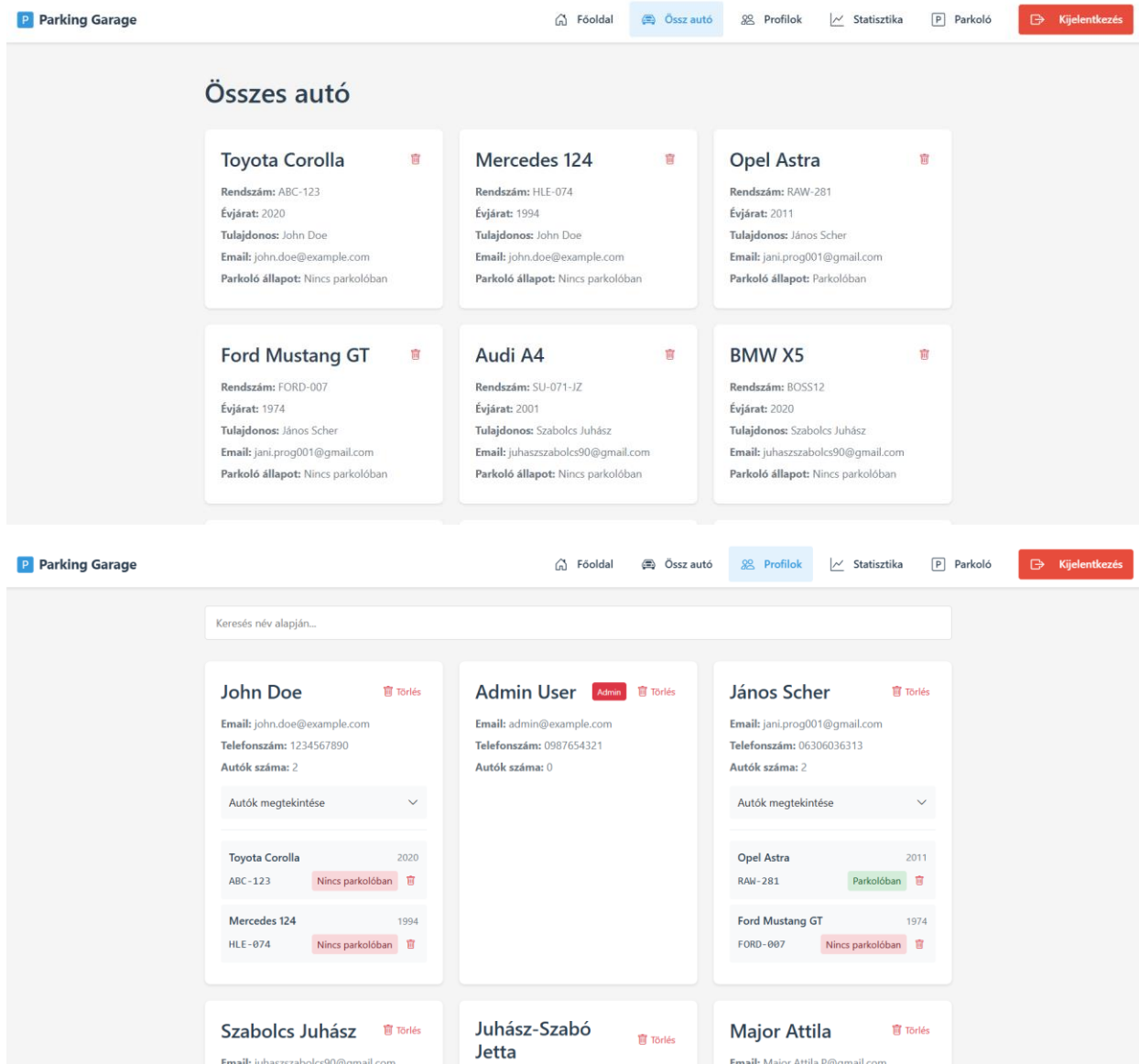
Összes parkolási idő

701 óra 25 perc

Átlagos parkolási díj

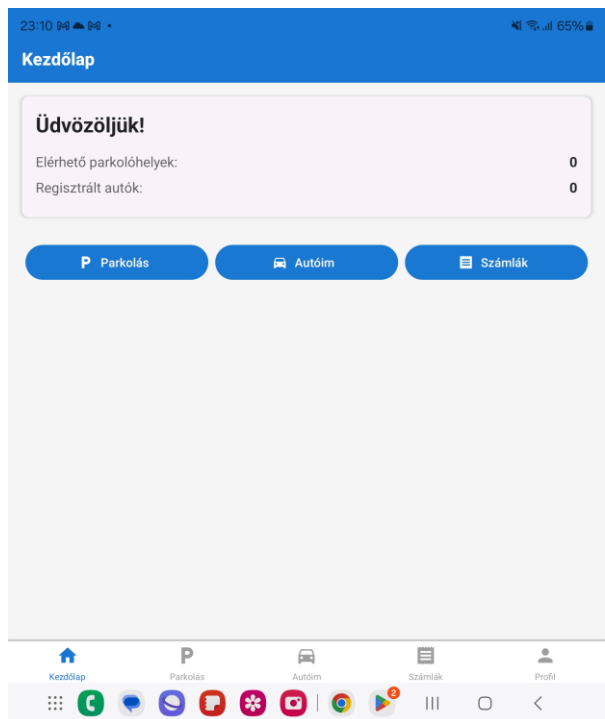
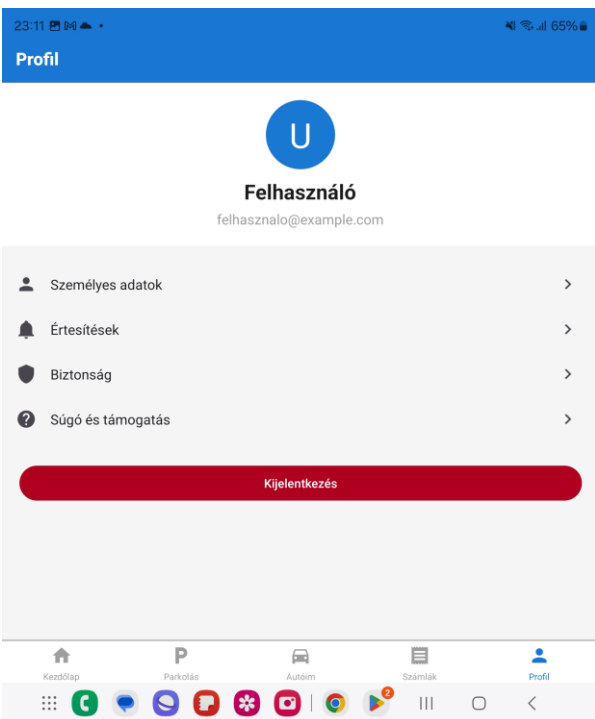
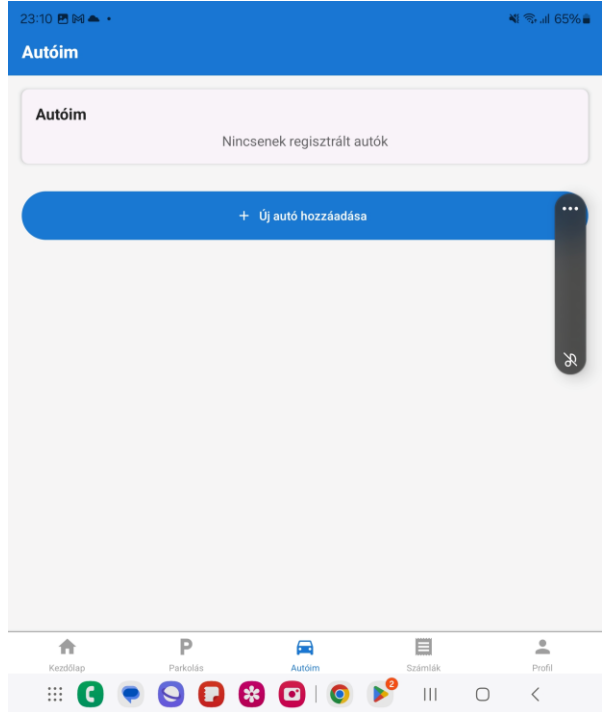
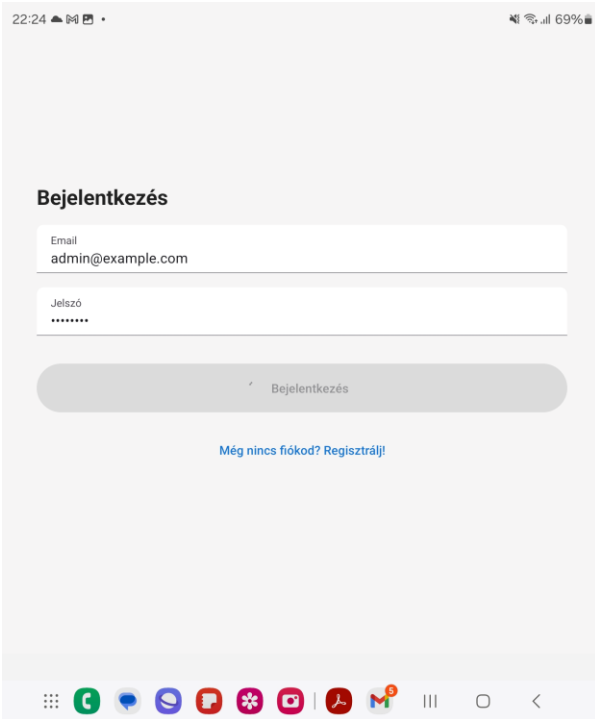
14033

Parkolóház © 2025



2. Mobil applikáció

- Egyszerű, átlátható felület
- Könnyű kezelhetőség



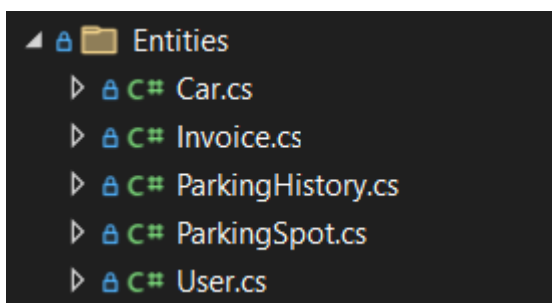
A felület főbb jellemzői:

- 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 mindig 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:



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)**

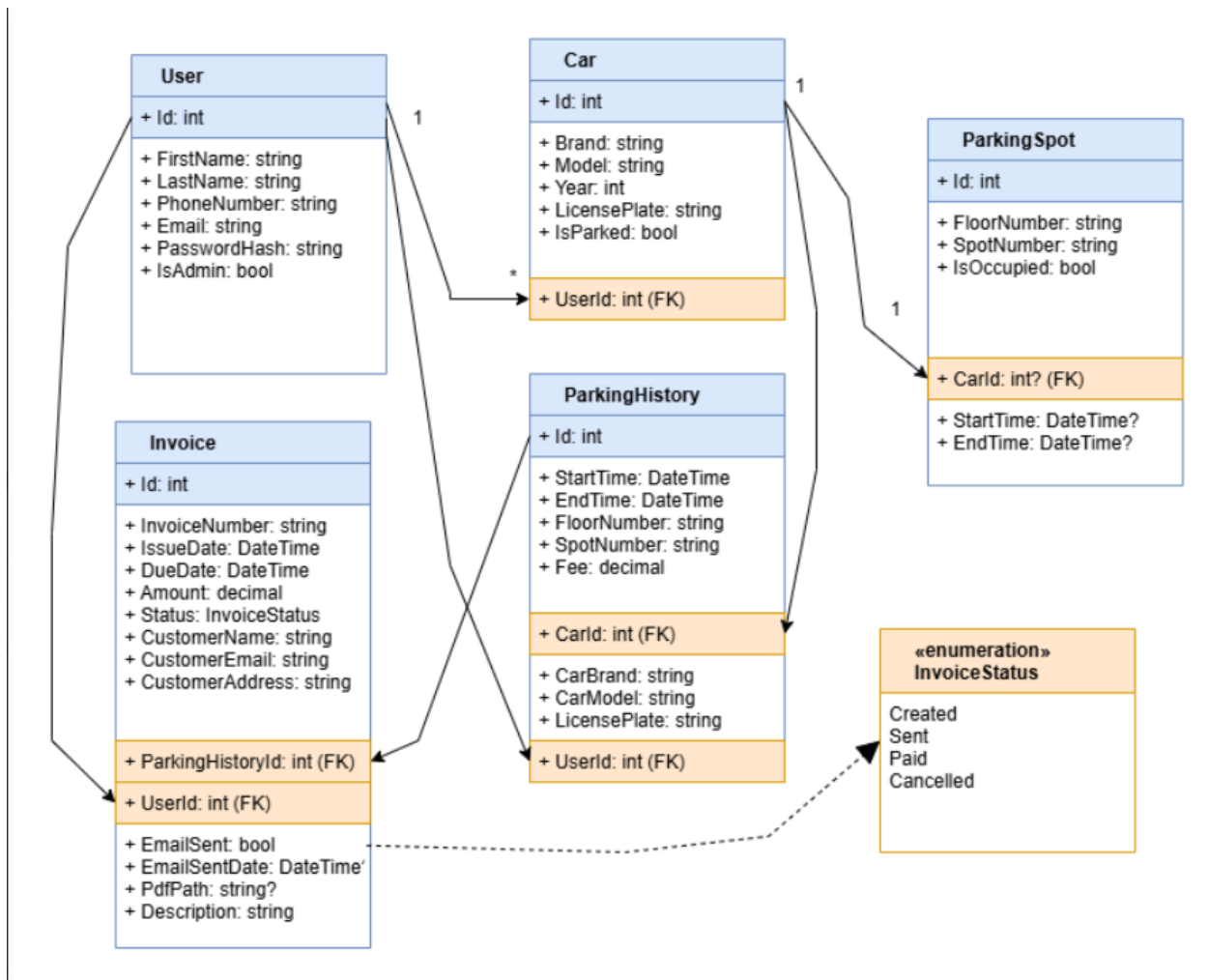
- Azonosító, emelet szám, helyszám, foglaltság
- Kapcsolatok: jármű, parkolási események

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



Fejlesztői Környezet

A projekt fejlesztéséhez a következő eszközöket és technológiákat használtuk:

1. Backend

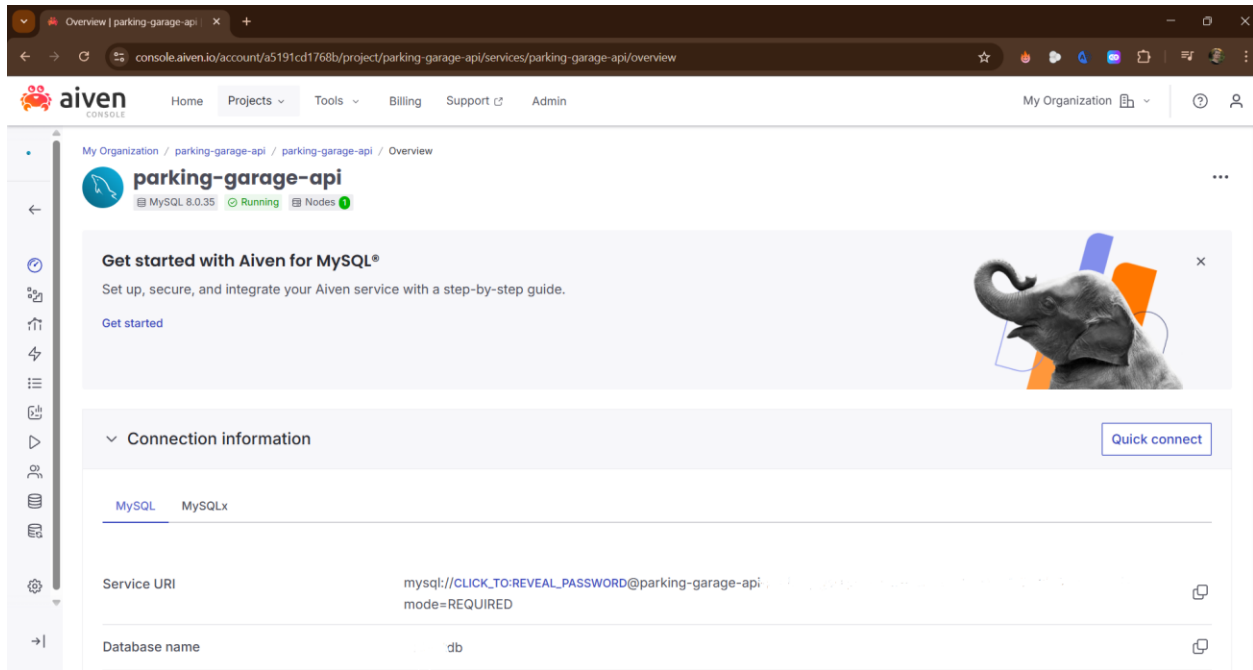
- ASP.NET Core 8.0
- Entity Framework Core 8.0
- Pomelo.EntityFrameworkCore.MySql
- Swagger/OpenAPI
- MailKit (email küldés)
- iTextSharp (PDF generálás)

2. Frontend

- Svelte

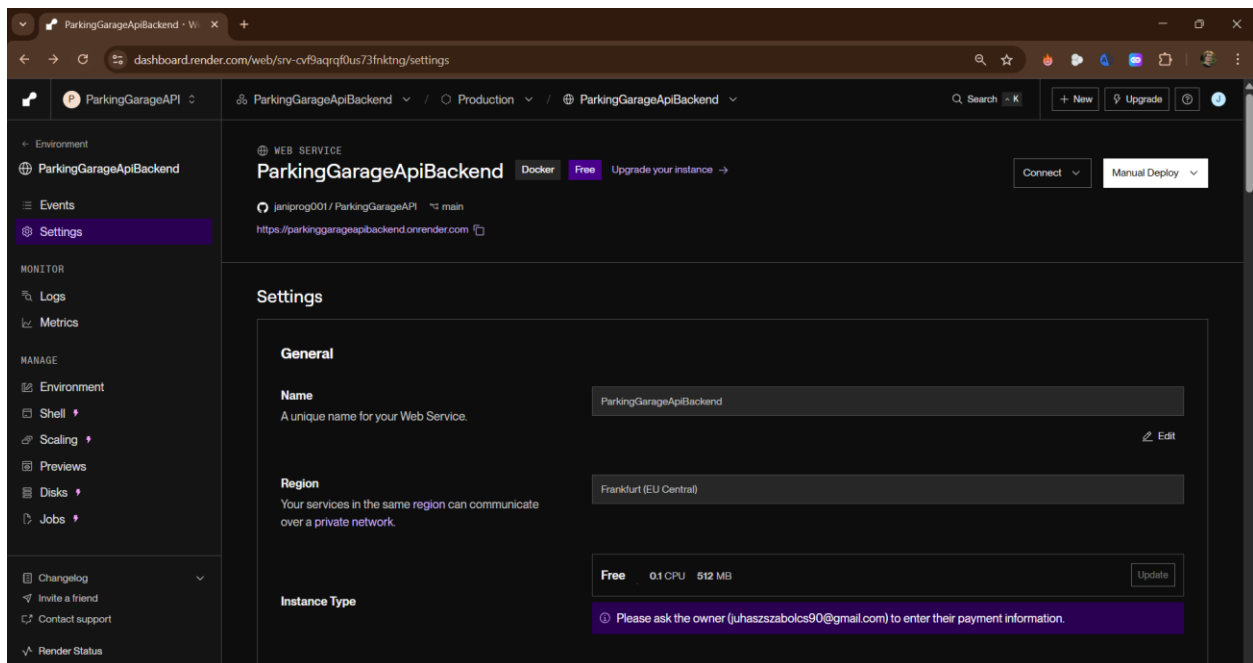
3. Adatbázis

- MySQL 8.0
- Aiven üzemeltetés



4. Deployment

- Render (backend)



ParkingGarageApiBackend · W · X +

← → ↻ dashboard.render.com/web/srv-cv9aqrg0us73fntking 🔍 ☆ 🔥 🗨️ 📁 📄 🌐 ⋮

ParkingGarageAPI ⌵ ⚙️ ParkingGarageApiBackend / ⌵ Production / ⚙️ ParkingGarageApiBackend 🔍 Search 🔑 + New ⚙️ Upgrade ⓘ

← Environment

⚙️ ParkingGarageApiBackend

☰ Events

⚙️ Settings

MONITOR

🔍 Logs

📊 Metrics

MANAGE

⚙️ Environment

📁 Shell

⚙️ Scaling

🖼️ Previews

📀 Disks

📁 Jobs

📄 Changelog

👤 Invite a friend

🗨️ Contact support

✓ Render Status

WEB SERVICE

ParkingGarageApiBackend Docker Free Upgrade your instance →

Connect Manual Deploy

janiprog001 / ParkingGarageAPI main
https://parkinggarageapibackend.onrender.com 📄

ⓘ Your free instance will spin down with inactivity, which can delay requests by 50 seconds or more. Upgrade now

✅ Deploy live for 624a17a: Merge pull request #6 from janiprog001/admin-autotories User statisztikák listázásának javítása
April 7, 2025 at 10:20 PM

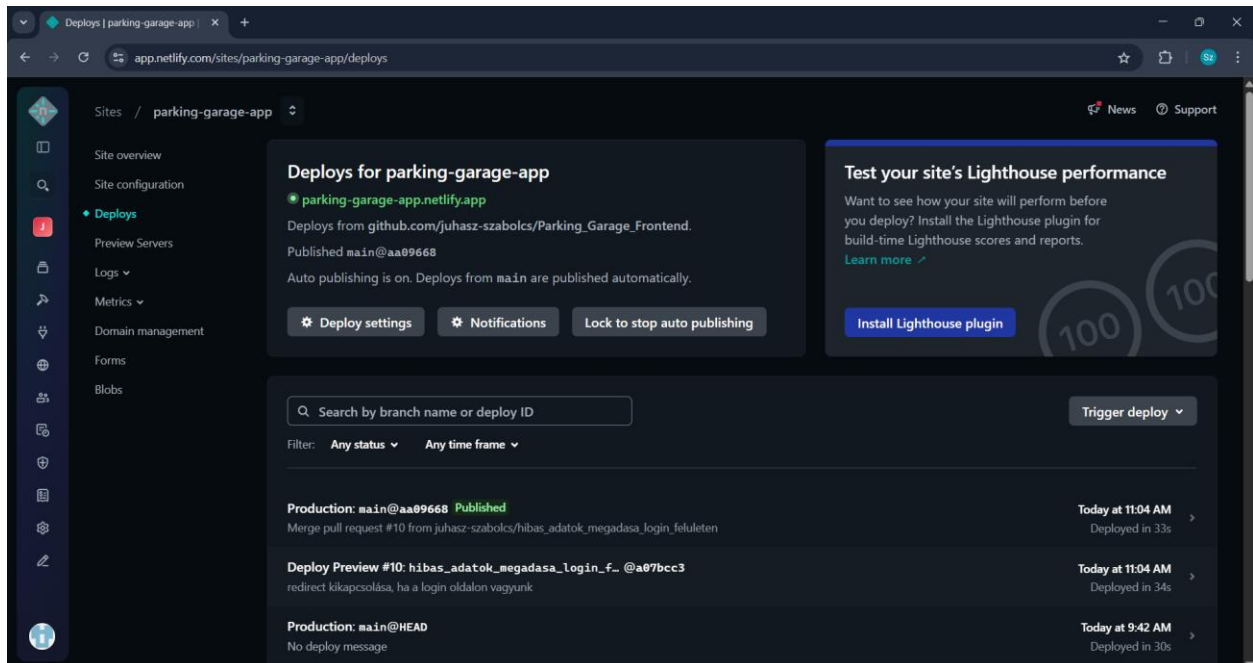
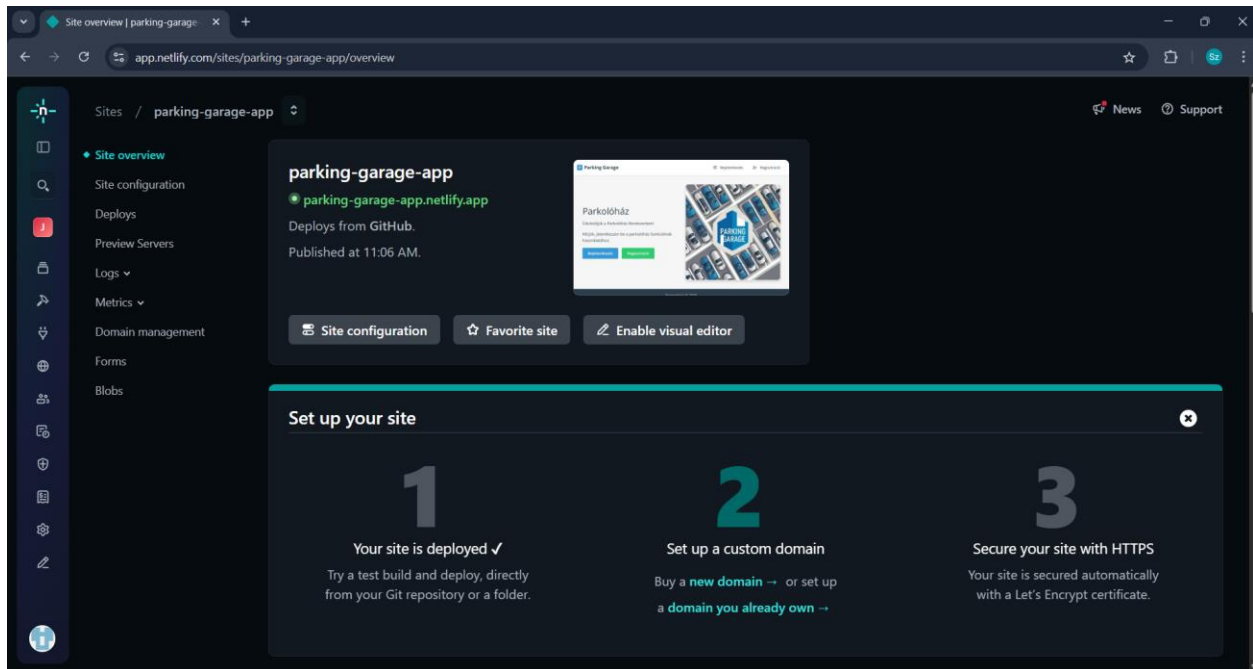
Deploy started for 624a17a: Merge pull request #6 from janiprog001/admin-autotories User statisztikák listázásának javítása
Manually triggered by you via Dashboard
April 7, 2025 at 10:19 PM

✅ Deploy live for 8ff1916: Merge pull request #5 from janiprog001/admin-autotories Admin autotories
April 7, 2025 at 5:35 PM Rollback

Deploy started for 8ff1916: Merge pull request #5 from janiprog001/admin-autotories Admin autotories
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 lekérése
April 5, 2025 at 10:47 AM Rollback

- Netlify (frontend)



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

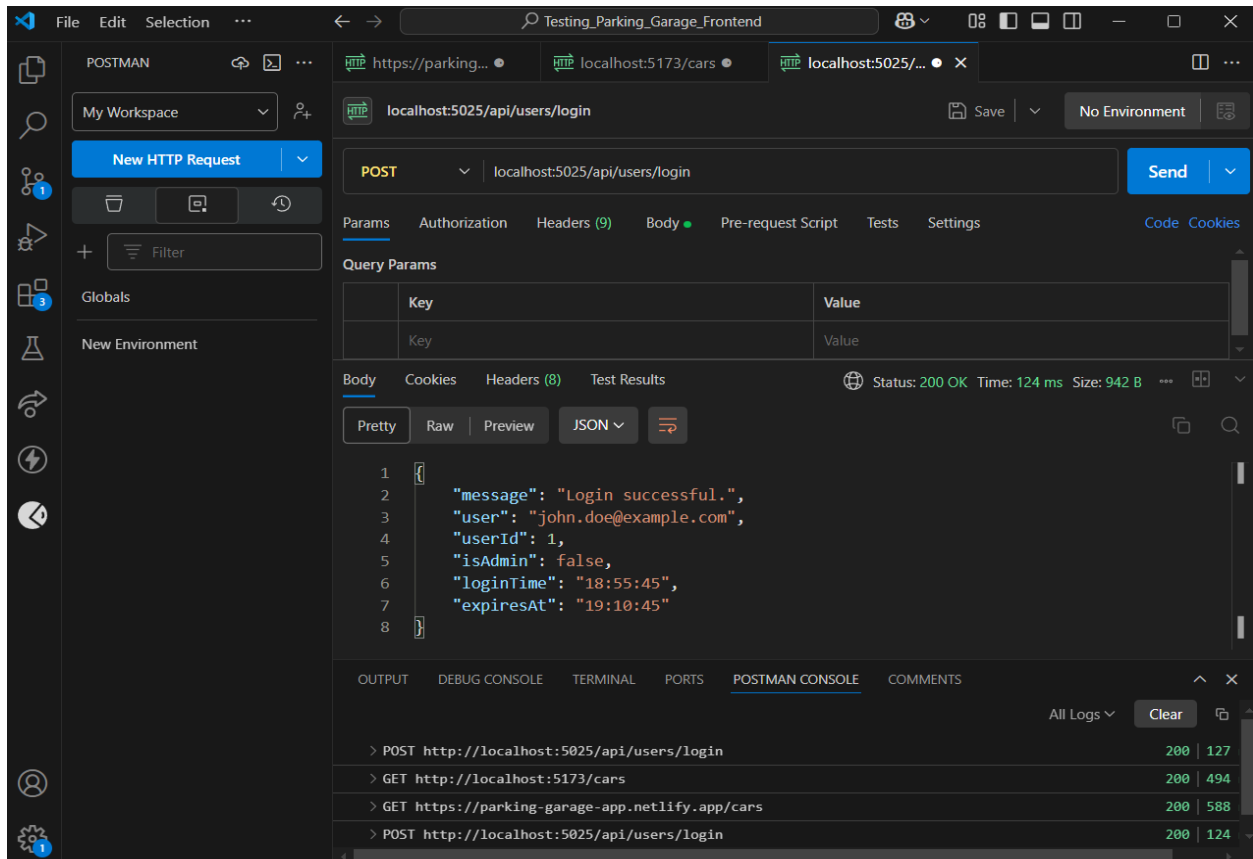
1. API Végpont Tesztelés

- Robot Framework végponttesztelés:

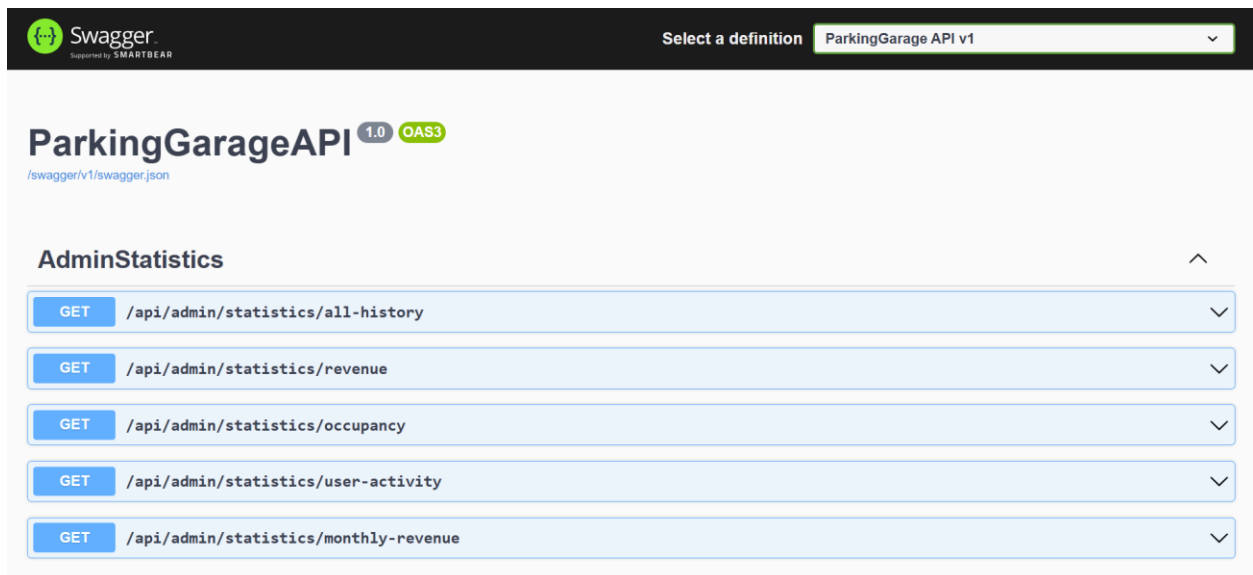
The screenshot displays a Robot Framework test report for a suite named 'Testing Parking Garage Frontend'. The report is structured as follows:

- SUITE: Testing Parking Garage Frontend**
 - Full Name: Testing Parking Garage Frontend
 - Source: Z:\Kiskunfelegyhaza\Testing_Parking_Garage_Frontend
 - Start / End / Elapsed: 20250414 19:34:10.597 / 20250414 19:34:35.759 / 00:00:25.162
 - Status: 1 test total, 1 passed, 0 failed, 0 skipped
 - SUITE: Testing Endpoints**
 - Full Name: Testing Parking Garage Frontend.Testing Endpoints
 - Source: Z:\Kiskunfelegyhaza\Testing_Parking_Garage_Frontend\testing_endpoints.robot
 - Start / End / Elapsed: 20250414 19:34:10.635 / 20250414 19:34:35.758 / 00:00:25.123
 - Status: 1 test total, 1 passed, 0 failed, 0 skipped
 - TEST: Test Login Endpoint Success**
 - Full Name: Testing Parking Garage Frontend.Testing Endpoints.Test Login Endpoint Success
 - Start / End / Elapsed: 20250414 19:34:11.443 / 20250414 19:34:35.757 / 00:00:24.314
 - Status: **PASS**
 - Keywords and Steps:
 - RequestLibrary.Create Session** parking_api \${BASE_URL} (00:00:00.001)
 - Buildin.Create Dictionary** Content-Type=application/json (00:00:00.001)
 - Buildin.Create Dictionary** email=\${VALID_EMAIL} password=\${VALID_PASSWORD} (00:00:00.001)
 - RequestLibrary.POST On Session** parking_api /api/users/login json=\${data} headers=\${headers} (00:00:24.293)
 - Buildin.Log** Response Status: \${response.status_code} (00:00:00.001)
 - Buildin.Log** Response Body: \${response.text} (00:00:00.001)
 - RequestLibrary.Status Should Be** 200 \${response} (00:00:00.000)
 - Buildin.Set Variable** \${response.json()} (00:00:00.001)
 - Collections.Dictionary Should Contain Key** \${response.json} userId (00:00:00.000)
 - Collections.Dictionary Should Contain Key** \${response.json} message (00:00:00.001)
 - Buildin.Should Be Equal** \${response.json["message"]} Login successful. (00:00:00.001)
 - Buildin.Get From Dictionary** \${response.json} userId (00:00:00.001)
 - Buildin.Set Global Variable** \${USER_ID} \${user_id} (00:00:00.002)

- Postman tesztelés



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



Car



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

Invoice



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

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}	▼

Statistics



GET

/api/statistics/history



GET

/api/statistics/summary



GET

/api/statistics/by-car



GET

/api/statistics/monthly



Test



GET

/api/test/userdata



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



Frontend Tesztelés

1. API Integráció Tesztelés

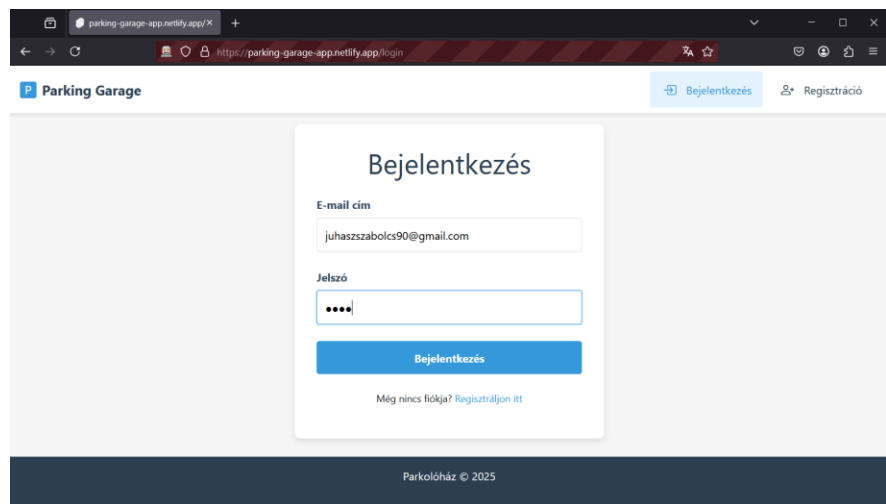
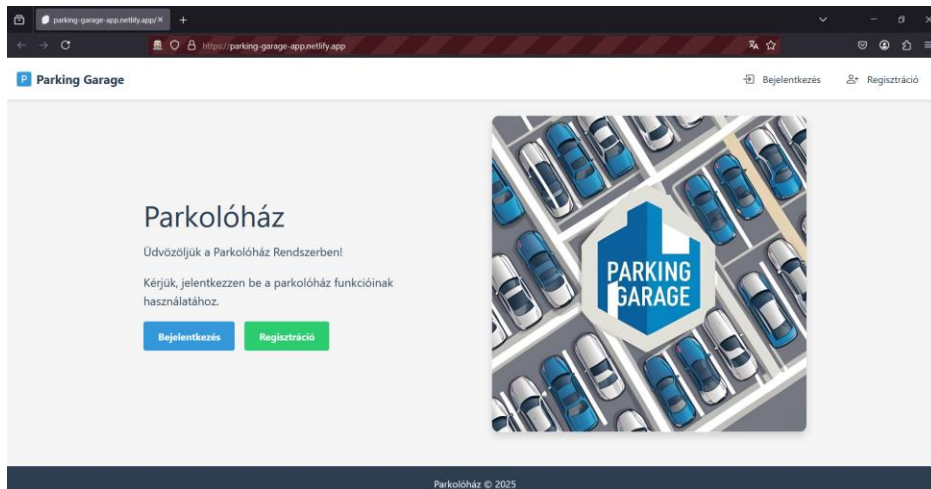
- Az api.js fájlban részletes hibakezelés és naplózás implementálva
- A kérések és válaszok részletes naplózása fejlesztési célokra
- CORS beállítások tesztelése különböző domainek között

2. Reszponzív Tesztelés

- A frontend részponzív 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. Teszteset létrehozása a bejelentkezésre Robot Framework-el:



Testing Parking Garage Frontend Report

Generated
20250414 20:03:43 UTC+02:00
7 minutes 2 seconds ago

Summary Information

Status: 1 test failed
Start Time: 20250414 20:02:04.728
End Time: 20250414 20:03:43.250
Elapsed Time: 00:01:38.522
Log File: [log.html](#)

Test Statistics

Total Statistics	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
All Tests	6	5	1	0	00:01:37	

Statistics by Tag	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
No Tags						

Statistics by Suite	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
Testing Parking Garage Frontend	6	5	1	0	00:01:39	
Testing Parking Garage Frontend . Testing Endpoints	3	2	1	0	00:00:04	
Testing Parking Garage Frontend . Testing Frontend	3	3	0	0	00:01:35	

Test Details

All Tags Suites Search

Suite:
Test:
Include:
Exclude:

Search Clear Help

RIDE - Testing Frontend

File Edit Tools Navigate Macros View Help

Test Suites

- Testing Parking Garage Frontend
 - Testing Endpoints
 - ① \${BASE_URL}
 - ② \${VALID_EMAIL}
 - ③ \${VALID_PASSWORD}
 - ✓ Test Login Endpoint Success
 - ✓ Test Login Endpoint Invalid
 - ✓ Test Get Cars Endpoint
 - ⚙ Login And Get UserId
 - Testing Frontend
 - ④ \${URL}
 - ⑤ \${VALID_EMAIL}
 - ⑥ \${INVALID_EMAIL}
 - ⑦ \${PASSWORD}
 - ⑧ \${USERNAME}
 - ✓ Login with valid credentials
 - ✓ Login with invalid email
 - ✓ Show cars
 - External Resources

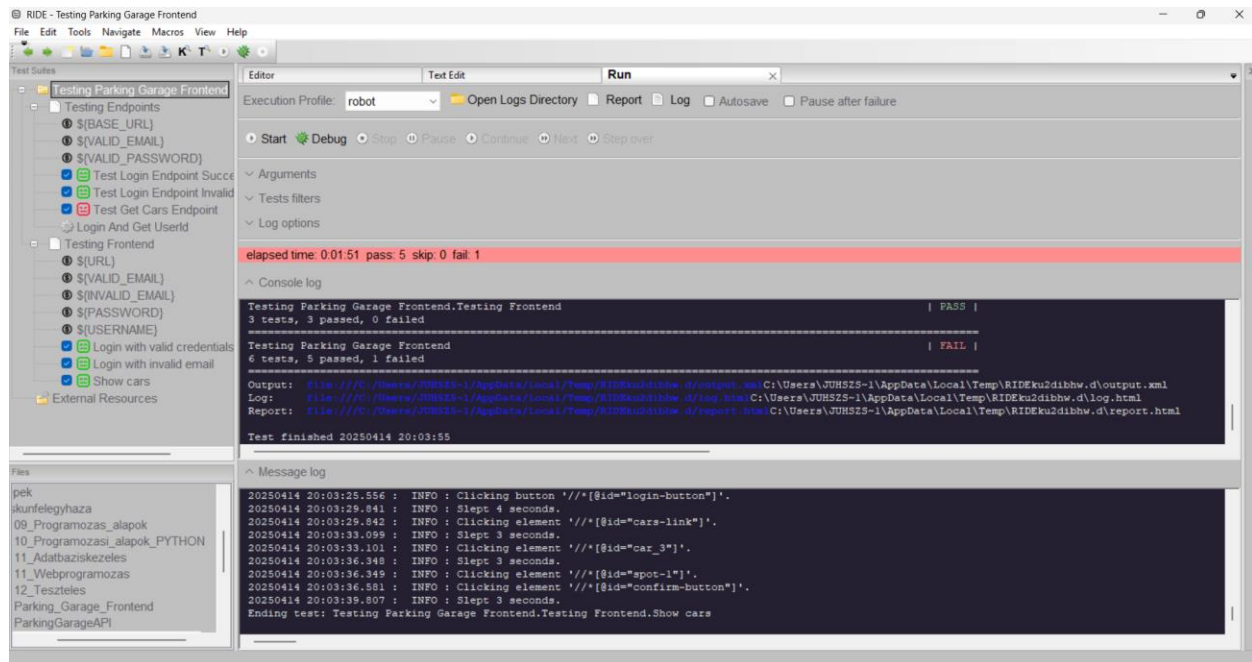
Editor

Text Edit Run

Login with invalid email

Settings

1	Open Browser	\$(URL)	Firefox	
2	Wait For Condition	return document.readyState == "complete"		
3	Click Element	//*[@id="login-link"]		
4	Wait Until Location Contains	/login	timeout=10s	
5	Input Text	//*[@id="email"]	\$(INVALID_EMAIL)	
6	Input Text	//*[@id="password"]	\$(PASSWORD)	
7	Click Button	//*[@id="login-button"]		
8	Get Selenium Implicit Wait			
9	Sleep	4		
10	Page Should Contain	Invalid email or password.		
11	Sleep	2s		
12	Close Browser			



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.

Felhasználói oldal

A felhasználók számára a rendszer **könnyen kezelhető és barátságos**:

- Regisztráció után azonnal hozzáadhatják járműveiket.
- A parkolás teljesen **automatizált**.
- A rendszer **automatikusan számolja a díjakat**, generálja a számlákat, és emailben el is küldi azokat.
- A felhasználók bármikor **megtekinthetik parkolási előzményeiket**.

Adminisztrációs oldal

Az adminisztrációs oldal a parkolóház üzemeltetői számára is **hatékony és átlátható kezelőfelületet** biztosít:

- Az adminok **kezelhetik a felhasználókat**, követhetik a **parkolóhelyek foglaltságát**, és **részletes statisztikákat** tekinthetnek meg.
- A számlák **automatikus generálása és küldése** jelentősen csökkenti az adminisztratív terheket.

Technikai megvalósítás

A rendszer modern technológiákra épül:

- **Backend**: ASP.NET Core 8.0, amely megbízható és gyors működést biztosít.
- **Frontend**: Svelte és TypeScript, amelyek reszponzív és kellemes felhasználói élményt nyújtanak.
- **Adatbázis**: MySQL, az **Aiven** szolgáltatáson keresztül hosztolva, amely **biztonságos és skálázható adattárolást** kínál.

A rendszer jelenleg sikeresen üzemel a **Render** (backend) és **Netlify** (frontend) platformokon, amelyek lehetővé tették a valódi környezetben történő tesztelést is. A felhasználói visszajelzések

alapján a rendszer **megbízhatóan működik**, és **valóban támogatja a parkolóházak üzemeltetését**.

További fejlesztési lehetőségek

A jövőben a következő fejlesztések jelenthetnek előrelépést:

- **Rendszerbeállítások felület** az adminok számára.
- **Audit log rendszer** a műveletek követhetősége érdekében.
- **Teljesítményoptimalizálás**.
- **További biztonsági fejlesztések**.

Összességében a projekt sikeresen demonstrálja, hogyan lehet **egy valós üzleti problémát megoldani modern technológiákkal**. A rendszer nemcsak működőképes, hanem valóban hasznosítható egy parkolóház üzemeltetésében, miközben a felhasználók számára is **kiváló élményt** nyújt.

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/>