Park1t&Go

Készítette:

Kovács Zétény

Tóth Bence

Zsiros Máté

**Nyilatkozat:**

**Kijelentjük, hogy a programkód teljes szövege, valamint a dokumentáció teljes egésze saját szellemi termékünk, melyet önállóan készítettünk.**

# Feladatspecifikáció

A célunk az volt, hogy egy könnyen használható parkolási platformot hozzunk létre. Ez nemcsak egy új lehetőséget ad, hanem leegyszerűsíti a parkolóházak kezelését és a parkolást.

A Park1t&Go többféle felhasználói szinttel rendelkezik: parkolóház-üzemeltetők, adminisztrátorok és parkolókat használó emberek. A cél az, hogy könnyebben lehessen foglalni és kezelni a parkolóhelyeket, valamint egyszerűbbé teszi az ügyfelek és üzemeltetők együttműködését.

Ez a rendszer nem csupán egy újabb alkalmazás, hanem segít az adminisztrációban, leegyszerűsíti a parkolást, és támogatja a környezetbarát megoldásokat, hozzájárulva a városi közlekedés jobb működéséhez.

# Alkalmazott szoftverek és technológiák

* 1. Alkalmazott szoftverek:
     + Visual Studio 2022 v17.13
     + Visual Studio Code v1.81
     + Android Studio v3.1.14
     + XAMPP v3.3.0
     + GitHub v3.16.0
     + Firefox verzió: 136.0.4
     + Git v2.49.0
     + Trello
     + Figma
     + Node.js v22.14.0
  2. Programnyelvek, keretrendszerek és modulok:

**Frontend:**

* + - * **Nyelv: JavaScript(x)**
      * **Keretrendszer: React**
      * **UI Komponenskönyvtár:**
      * **Routing:**

**Backend:**

* **Nyelv: JavaScript**
* **Web Framework: ReactJS**
* **Adatbázis kezelő: PhpMyAdmin**
* **Authentikáció és Authorizáció: JsonWebToken**
* **HTTP kliens:**
* **Middleware:**

**Android alkalmazás:**

* **Nyelv: Kotlin**
* **Keretrendszer: LadyBug 2022**

**Asztali alkalmazás:**

* **Nyelv: C#**
* **Keretrendszer: .Net**
  1. **Megvalósítás**

A parkolóház projektet otthoni környezetben, saját munkaeszközökön, engedélyezett tanórák keretében valósítottuk meg. Az asztali alkalmazást C# WinForm-ban fejlesztettük, az Androidos verziót Kotlin nyelven írtuk, míg a weboldalt React és JavaScript technológiával építettük fel. A három platform közötti adatáramlást egy központi RESTful API biztosítja, amely egységes kommunikációt tesz lehetővé a backend és a különböző felületek között.

A verziókövetést GitHub-on végeztük, a weboldal működését pedig a Google Chrome és a Mozilla Firefox legújabb verzióiban teszteltük. A projekt során külön kihívást jelentettek az új technológiák – például a React – gyors elsajátítása, valamint az egy főre jutó munkamennyiség, amely gondos tervezést és precíz kivitelezést igényelt a többplatformos megoldás kialakításában.

# A felület funkciói, felépítése és használata

A felületek felépítésénél a legfőbb szempontjai az átláthatóság és az egyszerűség voltak.

* 1. **Natív asztali alkalmazás**
* Rendszergazdai kezelőfelület.
* Felhasználók áttekintése, felhasználók adatainak módosítása, felhasználók törlése és létrehozása.
* Parkolóházak áttekintése, módosítása, törlése és hozzáadása.
* Foglalások áttekintése, módosítása és törlése. Abban az esetben, ha létezik a felhasználó és jövőbeni foglalást szeretne létrehozni, az adott felhasználó számára foglalás létrehozása.

**Asztali alkalmazás drótváza és prototípusa:**

A képen képernyőkép, Téglalap, tervezés, szöveg látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, Multimédiás szoftver, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

A képen szöveg, fa, képernyőkép, ég látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Az alkalmazás futtatása után egy főoldal fog megjelenni, melyen két gomb található, bejelentkezés és kilépés.

A kilépés gomb megnyomásával egy felugró ablak megkérdezi, hogy biztosan ki akarunk-e lépni, igen válaszra az alkalmazás leáll.

A képen szöveg, ég, kültéri, fa látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

A bejelentkezés gombra kattintva egy másik felület jelenik meg. Ezen a felületen tudja megadni a rendszergazdai jogosultsággal rendelkező felhasználó a felhasználónevét és a jelszavát.

**A képen szöveg, képernyőkép, Betűtípus, tervezés látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.**

Amennyiben létezik felhasználó ilyen névvel és megfelelő jogosultsággal, illetve jelszó is egyezik, felugró ablakon jelez vissza az alkalmazás a belépés sikerességéről.

A képen szöveg, képernyőkép, diagram, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

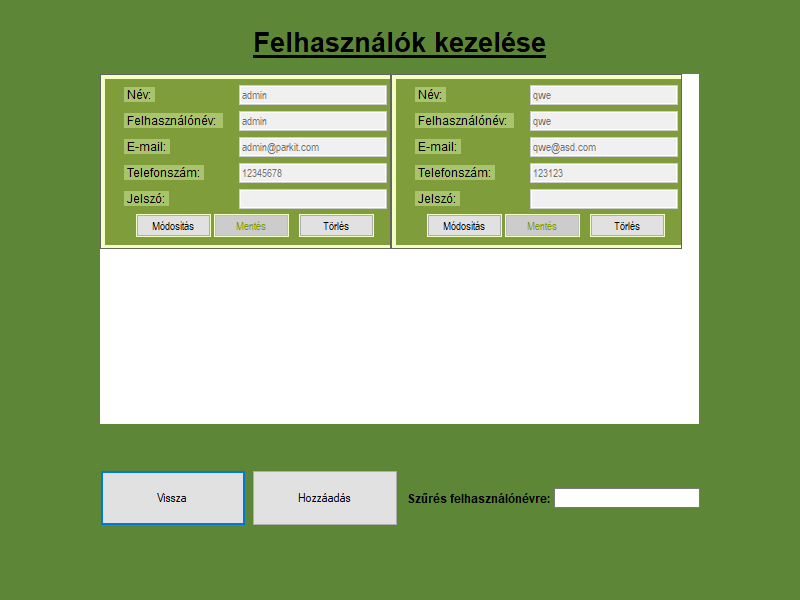
Abban az esetben, ha a bejelentkezés sikeres, a fő menü felülete megjelenik. A fő menüben 3 további választási opciót látunk. felhasználók kezelése, Parkolóházak kezelése és Foglalások kezelése, illetve egy kijelentkezés gombot.

****

A felhasználók kezelése gombra kattintva egy másik felület jelenik meg. Ezen a felületen tudjuk áttekinteni a regisztrált felhasználókat, néhány fontosabb adatukkal együtt.

A felületen található még egy szűrésre alkalmas mező, a keresett felhasználónevet beírva, csak azt a felhasználót tartja meg a felületen, amivel megegyezik a keresett felhasználónév.

A felületen továbbá található még egy Vissza és egy Hozzáadás gomb is. A vissza gombra kattintva a fő menübe tudunk visszalépni.

****

A hozzáadás gombra kattintva a Felhasználó hozzáadása felület jelenik meg.

Itt tudunk rendszergazdaként felhasználói vagy rendszergazdai profilt regisztrálni a szükséges adatok megadásával és a hozzáadás gombra kattintva.

A felületen van egy Vissza gomb is, rákattintva vissza tudunk lépni a Felhasználók kezelése felületre.

A képen szöveg, képernyőkép, Betűtípus, Téglalap látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

A fő menüből van lehetőségünk belépni a Parkolóházak kezelése felületre.

Ezen a felületen tudjuk áttekinteni a létező parkolóházunkat, néhány fontosabb adatukkal együtt.

A felületen található még egy szűrésre alkalmas mező, a keresett parkolóház nevét beírva, csak azt a parkolóházat tartja meg a felületen, amivel megegyezik a keresett név.

A felületen továbbá található még egy Vissza és egy Hozzáadás gomb is. A vissza gombra kattintva a fő menübe tudunk visszalépni.

A képen szöveg, képernyőkép, képernyő, szám látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

A hozzáadás gombra kattintva a Parkolóház hozzáadása felület jelenik meg.

Itt tudunk rendszergazdaként parkolóházat létrehozni a szükséges adatok megadásával és a hozzáadás gombra kattintva.

A felületen található egy Vissza gomb is, rákattintva vissza tudunk lépni a Parkolóházak kezelése felületre.



A fő menüből van lehetőségünk belépni a Foglalások kezelése felületre.

Ezen a felületen tudjuk áttekinteni az összes aktív és inaktív foglalást, néhány fontosabb adatukkal együtt.

A felületen található még egy szűrésre alkalmas mező, a keresett foglalás tulajdonosának azonosítóját beírva, csak azt a foglalást tartja meg a felületen, amivel megegyezik a keresett felhasználói azonosító.

A felületen továbbá található még egy Vissza és egy Hozzáadás gomb is. A vissza gombra kattintva a fő menübe tudunk visszalépni.

A képen szöveg, képernyőkép, szoftver, képernyő látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

A hozzáadás gombra kattintva a Foglalás hozzáadása felület jelenik meg.

Itt tudunk rendszergazdaként foglalást létrehozni a szükséges adatok megadásával majd a hozzáadás gombra kattintva.

A felületen található egy Vissza gomb is, rákattintva vissza tudunk lépni a Parkolóházak kezelése felületre.

A képen szöveg, képernyőkép, Betűtípus, Téglalap látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

A fő menüben található még egy kijelentkezés gomb is, rákattintva egy felugró ablakon megkérdezi az alkalmazás, hogy biztos ki akarunk-e jelentkezni.

A képen szöveg, képernyőkép, Betűtípus, Téglalap látható

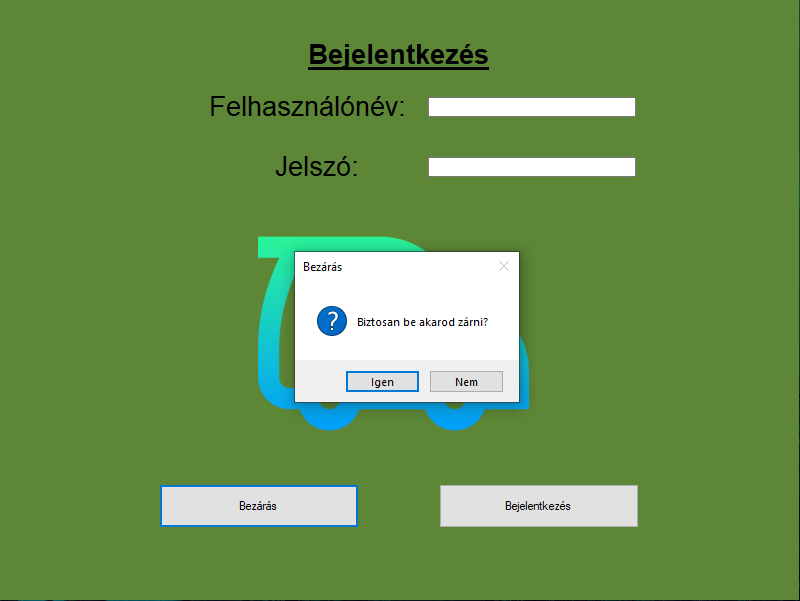
Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Az igenre kattintva az alkalmazás kijelentkeztet és a bejelentkezési felület jelenik meg.

A képen szöveg, képernyőkép, Betűtípus, diagram látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Innen a bezárás gombra kattintva egy felugró ablakon megkérdezi az alkalmazás, hogy biztos ki akarunk-e lépni. Ha az igenre nyomunk, az alkalmazás bezáródik.



* 1. **Androidos, mobil alkalmazás**
* **Mobilos felhasználói felület**
* **A felhasználó tud:** 
  + - **regisztrálni majd belépni a megadott adataival**
    - **személygépjárműve adatait megadni később módosítani**
    - **személyes adatait módosítani, és jelszót változtatni szükség esetén**
    - **parkolóházat és azon belül szabad parkolóhelyet választani (amennyiben nincs aktív foglalása és szabad a parkolóhely)**
    - **a jelenleg aktív foglalásait megnézni, ha van, és az eddigi foglalásait megtekinteni**

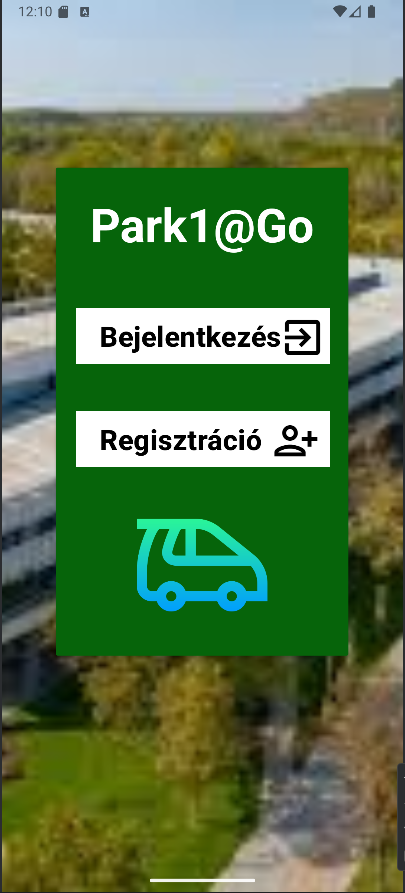
**Az androidos mobil alkalmazás képernyőterve és prototípusa:**

A képen képernyőkép, Téglalap, tervezés látható

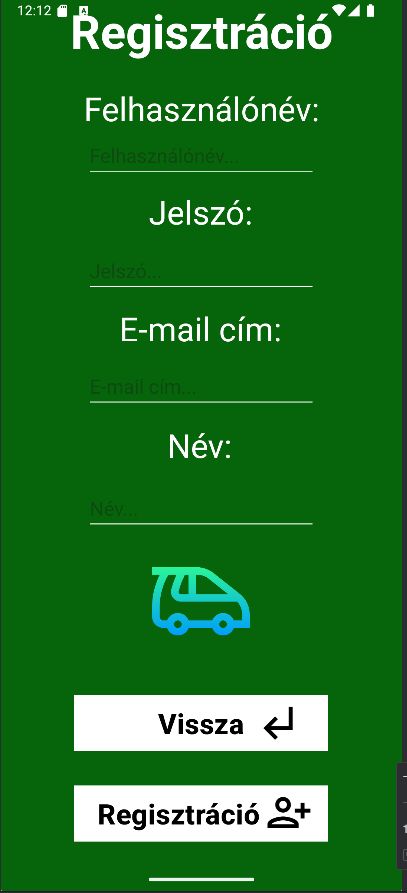
Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, Multimédiás szoftver, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Ha megnyitjuk a mobilos alkalmazást a főoldalt fogjuk látni, ahol a felhasználó két gombra tud rányomni: a ’Belépés’ és a ’Regisztráció’ gombok. ****

A ’Regisztráció’ gombra kattintva megjelenik a regisztrációs felület, ahol a felhasználó megtudja adni a megfelelő adatokat és regisztrálni azokkal. A regisztráció a ’Regisztrál’ gombbal történik, ha minden adat a formátumnak megfelelően ki van töltve. Ha nincs minden adat megfelelően megadva akkor ennek megfelelő hiba üzenetet kap. Amennyiben sikeresen regisztrált és újra rányom a regisztrációs gombra, akkor az a belépési felületre viszi őt. A ’Vissza’ gombbal a felhasználó vissza tud menni az előző nyitó oldalra.

****

Ha a felhasználó a belépési felületre jutott akkor ott a regisztrációban megadott adatokkal be tud jelentkezni, ha megnyomja a ’Belépés’ gombot. Ha a megadott adatok nem egyeznek akkor hibát kap a felhasználó, és nem tud belépni a felhasználói menübe. Az oldalon található még egy ’Regisztráció’ gomb, ha a felhasználó még nem regisztrált. Ez a gomb a regisztrációs felületre viszi.

****

Ha a felhasználó sikeresen bejelentkezett a ’Felhasználói menü’ felületre jut. Ezen az oldalon a felhasználó 5 gombra tud rányomni.



A ’Kocsim adatai’: ha rányom erre a felhasználó, egy olyan felület jelenik meg ahol a felhasználó személy gépjárművet tud hozzáadni (amennyiben nincs regisztrálva még az autója) vagy annak adatait módosítani (ha rendelkezik már regisztrált autóval). Új járművet az ’Autó regisztráció’ gombbal tud hozzáadni, és adatokat módosítani pedig az ’Jármű adatok módosítása’ gombbal tud. A ’Vissza’ gomb visszaviszi a felhasználót a ’Felhasználói menü’ felületre.



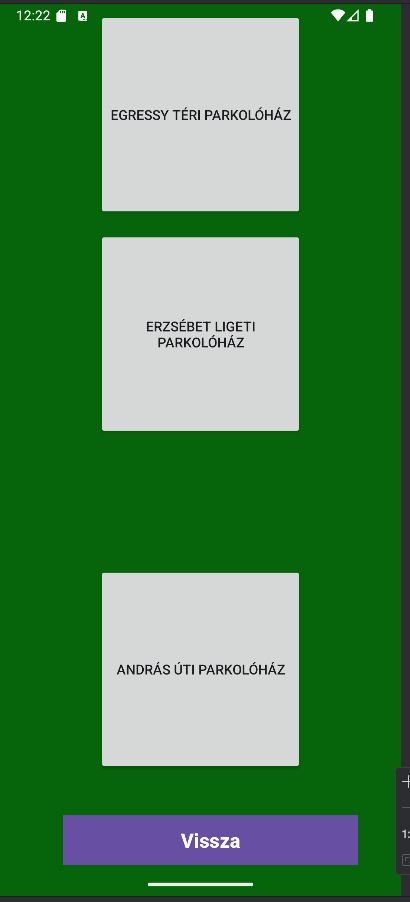
A ’Saját profil’: ezen a felületen a felhasználó, megtudja nézni az általa megadott adatokat, szükség eseten ezeket tudja módosítani az ’Adatok módosítása’ gombbal. Ezentúl, ha kell a felhasználó tudja módosítani a jelszavát is, erre egy külön felület van amire a ’Jelszót változtatok’ gomb viszi a felhasználót.

****

A jelszó változtató felületen lehetősége van a felhasználónak jelszót változtatni, abban az esetben, ha beírja a régi jelszavát, és kétszer az új jelszót. Ha valamelyik jelszó nem egyezik (például: a régi jelszó más, vagy az új jelszavak nem egyeznek) akkor ezt nem tudja megtenni. Ha minden egyezik és megfelelő akkor eltudja menteni az új jelszavát a ’Új jelszó mentése’ gombbal. A ’Vissza’ gombbal a felhasználó visszakerül a ’Saját profil’ felületre.

****

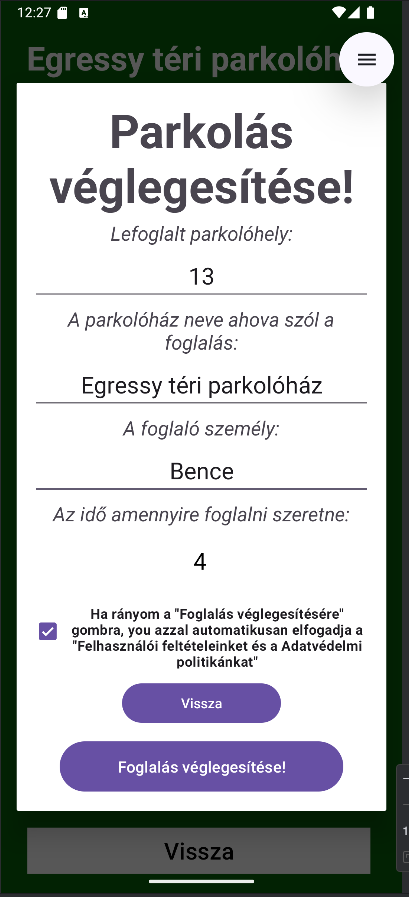
A ’Parkolóházak’: ezen a felületen a felhasználó ki tudja választani a számára tetszőleges parkolóházat annak megnyomásával. A ’Vissza’ gomb megnyomásával vissza kerül a felhasználó a ’Felhasználói menü’ felületre.



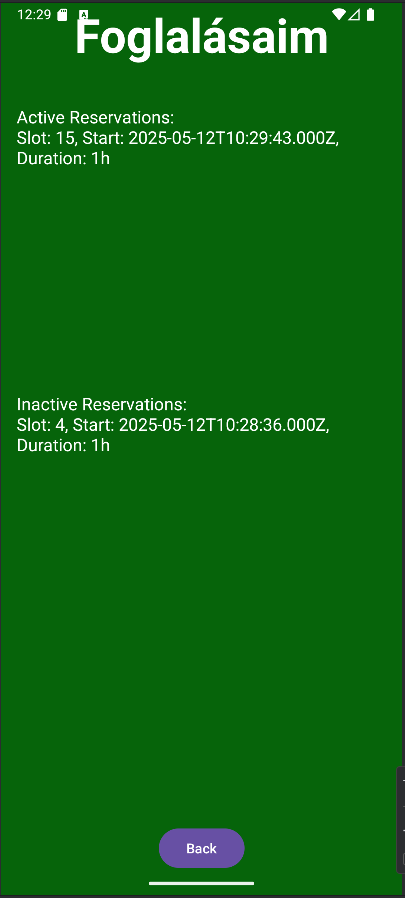
A felhasználó miután kiválasztotta a számára megfelelő parkolóházat, bekerül a ’Parkolóhely választó’ felületre. Ezen a felületen tud a felhasználó parkolóhelyet választani. A zöld színnel jelölt parkoló helyek szabadok ezeket tudja lefoglalni. A piros helyek, más felhasználók által lefoglalt parkolók, ezeket nem tudja választani a felhasználó. Ha a felhasználó rányom egy zöld helyre akkor az ’Parkolás véglegesítése’ felület nyílik meg.

****

A felhasználó a ’Parkolás véglegesítése’ felületen tudja véglegesíteni a foglalását. Ezen a felületen automatikusan megjelennek a fontos adatok, amiket a felhasználónak csak ellenőriznie kell. Miután a felhasználó ellenőrizte és elfogadta a felhasználói feltételeket, a foglalását a ’Foglalás véglegesítve’ gombbal tudja megtenni (amennyiben nincs aktív foglalása). Ezután a parkolóhely pirossá válik más felhasználóknak és nem tudja lefoglalni. A ’Vissza’ gombbal a felhasználó vissza kerül a ’Parkolóhely választó’ oldalra.

****

A ’Foglalásaim’: ezen a felületen a felhasználó megtudja nézni az aktív foglalásait amennyiben van aktív foglalása. Ha van már lejárt foglalása akkor azt is megtudja nézni. A ’Vissza’ gombbal a felhasználó vissza kerül a ’Felhasználói menü’ oldalra.

****

A ’Kijelentkezés’: ez egy gomb mellyel a felhasználó ki tud jelentkezni. A gomb megnyomásával vissza kerül a ’Bejelentkezés’ felületre.

* 1. **Reszponzív weboldal**

# Adatbázis

* 1. A megvalósítás egy lokális **MYSQL** szerverrel történt, **XAMPP** keretrendszerrel.

Használtuk a XAMPP által biztosított **Apache** szervert, **InnoDB** tárolómotorral.

A képen szöveg, képernyőkép, szoftver, szám látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

* 1. **Adatbázis kapcsolatok:**
     + - Car tábla:
         * Kapcsolatok:

Kapcsolódik a ’User’ táblához, az ’owner\_id’ mező egy felhasználóra vonatkozik.

* + - * + Egyedi azonosító:

A ’car\_id’ minden autót egyedileg azonosít.

* + - * + Tábla funkciója:

Autók adatainak (pl.: rendszám, magasság) eltárolása.

* + - * User tábla:
        + Kapcsolatok:

Kapcsolódik a ’Car’ táblához, a ’car\_id’ mező a felhasználó autójára vonatkozik.

Kapcsolódik a ’Reservation’ táblához, a ’reservation\_owner\_id’ jelöli a felhasználót, akihez a foglalás tartozik.

* + - * + Egyedi azonosító:

Az ’id’ minden felhasználót egyedileg azonosít.

* + - * + Tábla funkciója:

Felhasználói adatok (pl.: név, e-mail) eltárolása.

* + - * Parkhouse tábla:
        + Kapcsolatok:

Kapcsolódik a ’Reservation’ táblához, a ’park\_house\_id’ jelöli, hogy az adott foglalás melyik parkolóházhoz tartozik.

* + - * + Egyedi azonosító:

Az ’id’ minden parkolóházat egyedileg azonosít.

* + - * + Tábla funkciója:

Parkolóházak adatainak (pl.: kapacitás, nyitvatartás) eltárolása.

* + - * Reservation tábla:
        + Kapcsolatok:

Kapcsolódik a ’User’ táblához, a ’reservation\_owner\_id’ jelöli a felhasználót, akihez a foglalás tartozik.

Kapcsolódik a ’Parkhouse’ táblához, a ’park\_house\_id’ jelöli, hogy az adott foglalás melyik parkolóházhoz tartozik.

* + - * + Egyedi azonosító:

Az ’id’ minden foglalást egyedileg azonosít.

* + - * + Tábla funkciója:

Foglalások adatainak (pl.: állapot, időtartam) eltárolása.

* + 1. **Car tábla:**

A képen szöveg, képernyőkép, Betűtípus, szám látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

* **car\_id:** integer – Egyedi azonosító minden autóhoz.
* **plate:** varchar – Az autó forgalmi rendszáma.
* **height:** int – Az autó magassága centiméterben.
* **type:** varchar – Az autó márkajelzése.
* **owner\_id:** Az autó tulajdonosának az azonosítója.
  1. **User tábla:**

A képen szöveg, képernyőkép, Betűtípus, szám látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

* **id:** integer – Egyedi azonosító minden felhasználóhoz.
* **name:** varchar – A felhasználó neve.
* **car\_id:** integer – A felhasználó autójának azonosítója.
* **email:** varchar – A felhasználó e-mail címe.
* **phone\_num:** varchar – A felhasználó telefonszáma.
* **role:** varchar – A felhasználó szerepköre.
* **username:** varchar – A felhasználó felhasználóneve.
* **password:** varchar – A felhasználó jelszava.
  1. **Parkhouse tábla:**

A képen szöveg, képernyőkép, Betűtípus, szám látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

* **id:** integer – Egyedi azonosító minden parkolóházhoz.
* **name:** varchar – A parkolóház neve.
* **capacity:** integer – A parkolóház maximális férőhelye.
* **address:** varchar – A parkolóház címe.
* **rating:** varchar – A parkolóház értékelése.
* **opening:** datetime – A parkolóház nyitási dátuma.
* **closing:** datetime – A parkolóház zárási dátuma.
* **car\_height:** integer – A parkolóház autók befogadására képes maximális magassága.
* **max\_stay\_time:** integer – A parkolóházban tölthető maximális idő.
  1. **Reservation tábla:**

A képen szöveg, képernyőkép, multimédia, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

* **id:** integer – Egyedi azonosító minden foglaláshoz.
* **start\_time:** datetime – A foglalás kezdetének dátuma.
* **active:** integer – A foglalás aktív állapota (0 vagy 1).
* **inactive:** integer – A foglalás inaktív állapota (0 vagy 1).
* **reservation\_time\_hour:** integer – A foglalás maximális időtartama.
* **sum:** integer – A foglalás végösszege.
* **reservation\_owner\_id:** integer – A foglaló személy azonosítója.
* **park\_slot:** integer – A foglaláshoz tartozó parkolóhely.
* **park\_house\_id:** integer – A foglaláshoz tartozó parkolóház.

# API végpontok

* Rendszergazdai végpontok:
  + **Login**:
    - **Végpont**: /loginAdmin
    - **Metódus**: POST
    - **Leírás**: Ez a végpont a bejelentkezést biztosítja.
    - **Hibakezelés**:
      * Ha nincs a bejelentkezési adatoknak megfelelő felhasználó, 401-es hibakódot és 'Hibás felhasználónév, jelszó vagy nem megfelelő jogosultság.' hibaüzenetet küld vissza.
      * Ha a jelszó helytelen 401-es hibakódot és ’Hibás jelszó’ hibaüzenetet küld vissza.
      * Váratlan hiba esetén 500-as hibakódot és ’Bejelentkezési hiba.’ hibaüzenetet küld vissza.
  + Profilok lekérdezése:
    - **Végpont:** /profileAdmin
    - **Metódus:** GET
    - **Leírás:** Lekérdezi az összes felhasználó adatát.
    - **Autentikáció:** Érvényes JWT token szükséges, mivel az authenticate() middleware védi a hozzáférést.
    - **Válasz:** Sikeres lekérdezés esetén a rendszer az összes felhasználót tartalmazó tömböt ad vissza.
    - **Hibakezelés:**
      * Ha egyetlen felhasználó sem található, akkor 404-es hibakóddal tér vissza a rendszer, az üzenet: ’Nincs felhasználó!’.
      * Váratlan hiba esetén 500-as hibakódot küld vissza, az üzenet: ’Profil lekérdezési hiba!’.
  + Profiladatok frissítése:
    - **Végpont:** /profileDataUpdateAdmin/:id
    - **Metódus:** PUT
    - **Leírás:** Egy felhasználó adatainak frissítése.
    - **Autentikáció:** Érvényes JWT token szükséges.
    - **Paraméterek:**
      * • **Útvonalparaméter:**
        + id – a módosítandó felhasználó azonosítója.
      * • **Request Body-ban (opcionális mezők):**
        + username – új felhasználónév;
        + password – új jelszó (amely bcrypt algoritmussal kerül hash-elésre, ha meg van adva);
        + name – új név;
        + email – új e-mail cím;
        + phone\_num – új telefonszám.
    - **Válasz:** Sikeres frissítés esetén az üzenet "Adatok sikeresen frissítve" jelenik meg, valamint a módosított felhasználó adatai kerülnek visszaküldésre.
    - **Hibakezelés:**
      * Ha a megadott azonosítójú felhasználó nem található, 404-es hibakódot és "Felhasználó nem található!" hibaüzenetet küld vissza.
      * Frissítési hiba esetén 500-as hibakódot és "Hiba történt az adatok frissítésekor!" hibaüzenetet küld vissza.
  + Profil törlése:
    - **Végpont:** /userProfileDeleteAdmin/:id
    - **Metódus:** DELETE
    - **Leírás:** Egy adott azonosítóval rendelkező felhasználó törlése az adatbázisból.
    - **Autentikáció:** Érvényes JWT token szükséges.
    - **Paraméterek:**
      * **Útvonalparaméter:**
        + id – a törlendő felhasználó azonosítója.
    - **Válasz:** Sikeres törlés esetén az üzenet "Felhasználó sikeresen törölve!" jelenik meg.
    - **Hibakezelés:**
      * Ha a törlendő felhasználó nem található, 404-es hibakódot és "Felhasználó nem található!" hibaüzenetet küld vissza.
  + Profil regisztrációja:
    - **Végpont:** /registerUser
    - **Metódus:** POST
    - **Leírás:** Új felhasználó létrehozása. A rendszer ellenőrzi, hogy az adott e-mail címmel ne létezzen már felhasználó, majd a beérkező jelszót bcrypt algoritmussal hash-elés után elmenti, illetve automatikusan generálja az új car\_id és reservation\_id értékeket.
    - **Autentikáció:** Érvényes JWT token szükséges.
    - **Paraméterek:**
      * **Request Body-ban várható mezők:**
        + registerUser – a regisztrálandó felhasználónév;
        + registerPassword – a jelszó;
        + registerEmail – a felhasználó e-mail címe;
        + registerRole – a felhasználó szerepköre (numerikus érték);
        + registerName – a felhasználó neve;
        + registerPhone – (opcionális) telefonszám; ha nincs megadva, az alapértelmezett érték "12345678".
    - **Válasz:** Sikeres regisztráció esetén 201-es állapotkóddal a rendszer egy JWT tokent küld vissza.
    - **Hibakezelés:**
      * Ha már létezik felhasználó az adott e-mail címmel, 409-es hibakódot és "Már létezik felhasználó ilyen e-mail címmel!" hibaüzenetet küld vissza.
      * Létrehozási hiba esetén 500-as hibakódot és "Hiba történt a felhasználó létrehozásakor." hibaüzenetet küld vissza.
  + Foglalások lekérdezése:
    - **Végpont:** /reservationAdmin
    - **Metódus:** GET
    - **Leírás:** Lekérdezi az összes foglalást.
    - **Autentikáció:** Érvényes JWT token szükséges.
    - **Válasz:** Siker esetén a rendszer az összes foglalást tartalmazó tömböt adja vissza.
    - **Hibakezelés:**
      * Ha nincs foglalás, 404-es hibakódot és "Nincs foglalás!" hibaüzenetet küld vissza.
      * Lekérdezési hiba esetén 500-as hibakódot és "Foglalás lekérdezési hiba!" hibaüzenetet küld vissza.
  + Foglalások frissítése:
    - **Végpont:** /reservationUpdateAdmin/:id
    - **Metódus:** PUT
    - **Leírás:** Egy adott foglalás adatainak módosítása.
    - **Autentikáció:** Érvényes JWT token szükséges.
    - **Paraméterek:**
      * **Útvonalparaméter:**
        + id – a frissítendő foglalás azonosítója.
      * **Request Body-ban (opcionális mezők):**
        + active – a foglalás aktív státusza;
        + sum – a fizetendő összeg;
        + reservation\_owner\_id – a foglalás tulajdonosának azonosítója;
        + park\_slot – a foglalni kívánt parkolóhely sorszáma;
        + parkhouse\_id – a parkolóház azonosítója.
    - **Válasz:** Sikeres frissítés esetén a rendszer az "Foglalás sikeresen frissítve" üzenettel együtt visszaküldi a frissített foglalási adatokat.
    - **Hibakezelés:**
      * Ha a módosítandó foglalás nem található, 404-es hibakódot és "Foglalás nem található!" hibaüzenetet küld vissza.
      * Frissítési hiba esetén 500-as hibakódot és "Hiba történt az foglalás frissítésekor!" hibaüzenetet küld vissza.
  + Foglalások törlése:
    - **Végpont:** /reservationDeleteAdmin/:id
    - **Metódus:** DELETE
    - **Leírás:** Egy adott foglalás törlése az adatbázisból.
    - **Autentikáció:** Érvényes JWT token szükséges.
    - **Paraméterek:**
      * **Útvonalparaméter:**
        + id – a törlendő foglalás azonosítója.
    - **Válasz:** Sikeres törlés esetén a rendszer az "Sikeres törlés!" üzenetet adja vissza.
    - **Hibakezelés:**
      * Ha a törlendő foglalás nem található, 404-es hibakódot és "Foglalás nem található!" hibaüzenetet küld vissza.
  + Foglalás létrehozása:
    - **Végpont:** /registerReservationAdmin
    - **Metódus:** POST
    - **Leírás:** Új foglalás létrehozása. Először ellenőrzi, hogy az adott parkolóházban a megadott parkolóhelyen ne legyen már aktív foglalás, majd létrehozza az új foglalást. A foglalás időtartamából a rendszer kiszámolja a fizetendő összeget.
    - **Autentikáció:** Érvényes JWT token szükséges.
    - **Paraméterek:**
      * **Request Body-ban (opcionális mezők):**
        + start\_time – a foglalás kezdési ideje;
        + resTime – a foglalás időtartama órában;
        + resOwnId – a foglalást végző felhasználó azonosítója;
        + slotId – a foglalni kívánt parkolóhely azonosítója;
        + parkhouseId – a parkolóház azonosítója.
    - **Válasz:** Sikeres létrehozás esetén a rendszer 201-es státusszal az "Foglalás sikeresen létrehozva!" üzenetet és a létrehozott foglalási adatokat küldi vissza.
    - **Hibakezelés:**
      * Ha az adott parkolóhely foglalt, 409-es hibakódot és "A parkolóhely foglalt!" hibaüzenetet küld vissza.
      * Létrehozási hiba esetén 500-as hibakódot és "Hiba történt a foglalás létrehozásakor." hibaüzenetet küld vissza.
  + Parkolóházak lekérdezése:
    - **Végpont:** /parkhouseAdmin
    - **Metódus:** GET
    - **Leírás:** Lekéri az összes parkolóház adatait.
    - **Autentikáció:** Érvényes JWT token szükséges.
    - **Válasz:** Sikeres lekérdezés esetén a rendszer egy tömbben adja vissza a parkolóházak adatait.
    - **Hibakezelés:**
      * Ha egyetlen parkolóház sem található, 404-es hibakódot és "Nincs létező parkolóházad!" hibaüzenetet küld vissza.
      * Lekérdezési hiba esetén 500-as hibakódot és "Foglalás lekérdezési hiba!" hibaüzenetet küld vissza.
  + Parkolóházak frissítése:
    - **Végpont:** /parkhousesUpdateAdmin/:id
    - **Metódus:** PUT
    - **Leírás:** Egy adott parkolóház adatainak módosítása.
    - **Autentikáció:** Érvényes JWT token szükséges.
    - **Paraméterek:**
      * **Útvonalparaméter:**
        + id – a frissítendő parkolóház azonosítója.
      * **Request Body-ban (opcionális mezők):**
        + name – új név;
        + capacity – új kapacitás érték;
        + address – új cím;
        + opening – nyitvatartás kezdete;
        + closing – nyitvatartás vége.
    - **Válasz:** Sikeres frissítés esetén a rendszer az "Parkolóház sikeresen frissítve" üzenetet, valamint a módosított parkolóház adatait adja vissza.
    - **Hibakezelés:**
      * Ha a megadott parkolóház nem található, 404-es hibakódot és "Parkolóház nem található!" hibaüzenetet küld vissza.
      * Frissítési hiba esetén 500-as hibakódot és "Hiba történt a parkolóház frissítésekor!" hibaüzenetet küld vissza.
  + Parkolóházak törlése:
    - **Végpont:** /parkhouseDeleteAdmin/:id
    - **Metódus:** DELETE
    - **Leírás:** Egy adott parkolóház törlése az adatbázisból.
    - **Autentikáció:** Érvényes JWT token szükséges.
    - **Paraméterek:**
      * **Útvonalparaméter:**
        + id – a törlendő parkolóház azonosítója.
    - **Válasz:** Sikeres törlés esetén a rendszer az "Sikeres törlés!" üzenetet küldi vissza.
    - **Hibakezelés:**
      * Ha a törlendő parkolóház nem található, 404-es hibakódot és "Parkolóház nem található!" hibaüzenetet küld.
  + Parkolóházak létrehozása:
    - **Végpont:** /registerParkhouseAdmin
    - **Metódus:** POST
    - **Leírás:** Ez az endpoint új parkolóház létrehozását teszi lehetővé adminisztrátori jogosultsággal. A rendszer először ellenőrzi, hogy a megadott név és cím alapján már létezik-e egy parkolóház. Ha nem talál ilyen rekordot, az adatbázisban létrejön egy új parkolóház, ahol az alapértelmezett érték a rating esetében 0.
    - **Autentikáció:** Érvényes JWT token szükséges.
    - **Bemeneti paraméterek:**
      * A kérés törzsében várható értékek:
        + pName: A parkolóház neve;
        + pCapacity: A parkolóház kapacitása;
        + add: A parkolóház címe;
        + open: A parkolóház nyitási ideje;
        + close: A parkolóház zárási ideje;
        + cHeight: A járművek számára megengedett maximális magasság;
        + maxstay: A parkolóházban megengedett maximális parkolási idő.
    - **Válasz:** Sikeres létrehozás esetén 201-es státuszkóddal az üzenet "Parkolóház sikeresen létrehozva!" jelenik meg, és a rendszer visszaküldi a létrehozott parkolóház adatait.
    - **Hibakezelés:**
      * Ha a bejövő adatoknak megfelelő parkolóház már létezik, 409-es hibakóddal és "Ez a parkolóház már létezik!" hibaüzenetet küld vissza.
      * Egyéb hiba esetén 500-as hibakóddal és "Hiba történt a parkolóház létrehozásakor." hibaüzenetet küld vissza.
* Felhasználói végpontok:
  + Regisztráció:
    - **Végpont:** /register
    - **Metódus:** POST
    - **Leírás:** Új felhasználó regisztrációját kezeli a rendszer. A bemeneti adatok között kötelezőként szerepel a regisztrációs felhasználónév, jelszó és e-mail cím. A rendszer ellenőrzi, hogy a felhasználónév vagy e-mail cím ne ismétlődjön meg, majd a jelszót hash-eli, és automatikusan generálja az új rekordhoz szükséges car\_id és reservation\_id értékeket.
    - **Paraméterek:**
      * **Request Body-ban várható mezők:**
        + registerUser – regisztrálandó felhasználónév;
        + registerPassword – regisztrációs jelszó;
        + registerEmail – felhasználó e-mail címe;
        + registerName – (opcionális) felhasználó neve;
        + registerPhone – (opcionális) telefonszám (ha nincs megadva, alapértelmezett "12345678" kerül felhasználásra).
    - **Válasz:** Sikeres regisztráció esetén 201-es státuszkóddal a rendszer egy tokennel reagál.
    - **Hibakezelés:**
      * Ha hiányosak az adatok, 400-as hibakódot és "Hiányzó adat a regisztrációs kérésben!" hibaüzenetet küld vissza.
      * Ha már létezik ilyen felhasználónév vagy email cím, 400-as hibakódot és "Már létezik ilyen felhasználónév vagy email cím!" hibaüzenetet küld vissza.
      * Regisztrációs hiba esetén 500-as hibakódot és "Regisztrációs hiba!" hibaüzenetet küld vissza.
  + Bejelentkezés:
    - **Végpont:** /login
    - **Metódus:** POST
    - **Leírás:** A felhasználók bejelentkezését kezeli. A felhasználó által megadott bejelentkezési név alapján a rendszer lekéri az adatbázisból a felhasználót, majd a bcrypt segítségével ellenőrzi a jelszót.
    - **Autentikáció:** Nem szükséges, mivel ez a bejelentkezési folyamat része (loginLimiter alkalmazva a brute force támadások elkerülése érdekében).
    - **Paraméterek:**
      * **Request Body-ban várható mezők:**
        + loginUser – bejelentkezési felhasználónév;
        + loginPassword – bejelentkezési jelszó.
    - **Válasz:** Sikeres bejelentkezés esetén a rendszer a tokent és a felhasználó azonosítóját küldi vissza.
    - **Hibakezelés:**
      * Ha a felhasználó nem található vagy nincs jelszó, 401-es hibakódot és "Hibás felhasználónév vagy jelszó" hibaüzenetet küld vissza.
      * Ha a jelszó érvénytelen, 401-es hibakódot és "Hibás felhasználónév vagy jelszó" hibaüzenetet küld vissza.
      * Bejelentkezési hiba esetén 500-as hibakódot és "Bejelentkezési hiba!" hibaüzenetet küld vissza.
  + Parkolóház kapacitás lekérdezése:
    - **Végpont:** /parkhouse/capacity/:id
    - **Metódus:** GET
    - **Leírás:** A rendszer az adott parkolóház azonosítója alapján lekéri a parkolóház kapacitását; ez segít megállapítani, hogy a parkolóház hány autó befogadására képes.
    - **Autentikáció:** Nem szükséges, mivel ez a végpont nyilvános.
    - **Paraméterek:**
      * Útvonalparaméter:
        + id – az adott parkolóház egyedi azonosítója.
    - **Válasz:** Sikeres lekérdezés esetén a rendszer visszaadja a parkolóház kapacitását, amely az "capacity" kulcs alatt kerül visszaadásra.
    - **Hibakezelés:**
      * Ha a megadott azonosítójú parkolóház nem található, 404-es hibakóddal és "Parkolóház nem található" hibaüzenettel küld vissza;
      * Ha hiba lép fel a lekérdezés során, 500-as hibakóddal és "Hiba történt a parkolóház kapacitás lekérdezése során" hibaüzenettel reagál.
  + Profil lekérdezése:
    - **Végpont:** /profile
    - **Metódus:** GET
    - **Leírás:** A rendszer a bejelentkezett felhasználó azonosítását követően lekéri az adott felhasználó adatait az adatbázisból. A jelszó mezőt kizárja a visszaküldött adatokból.
    - **Autentikáció:** Érvényes JWT token szükséges az authenticate middleware miatt.
    - **Paraméterek:**
      * A felhasználó azonosítója a tokenből kerül meghatározásra; nem várunk további URL vagy query paramétereket.
    - **Válasz:** Siker esetén a rendszer a lekérdezett felhasználó adatait adja vissza.
    - **Hibakezelés:**
      * Ha a felhasználó nem található, 404-es hibakódot és "Felhasználó nem található!" hibaüzenetet küld vissza.
      * Ha hiba történik a lekérdezés során, 500-as hibakódot és "Profil lekérdezési hiba!" hibaüzenetet küld vissza.
  + Profil adatainak frissítése:
    - **Végpont:** /profileDataUpdate
    - **Metódus:** PUT
    - **Leírás:** A bejelentkezett felhasználó módosíthatja saját adatait. A rendszer a tokenből kapott azonosító alapján keresi meg a felhasználót, és a beérkező értékekkel (opcionális módosítások) frissíti a rekordot.
    - **Autentikáció:** Érvényes JWT token szükséges.
    - **Paraméterek:**
      * **Request Body-ban várható mezők:**
        + username – új felhasználónév;
        + password – új jelszó (amely bcrypt algoritmussal kerül hash-elésre, ha meg van adva);
        + name – új név;
        + email – új e-mail cím;
        + phone\_num – új telefonszám.
    - **Válasz:** Sikeres frissítés esetén az "Adatok sikeresen frissítve" üzenet jelenik meg, valamint a frissített felhasználói adatok kerülnek visszaküldésre.
    - **Hibakezelés:**
      * Ha a módosítandó felhasználó nem található, 404-es hibakódot és "Felhasználó nem található!" hibaüzenetet küld vissza.
      * Frissítési hiba esetén 500-as hibakódot és "Hiba történt az adatok frissítésekor!" hibaüzenetet küld vissza.
  + Autó hozzáadása:
    - **Végpont:** /registerCar
    - **Metódus:** POST
    - **Leírás:** Egy bejelentkezett felhasználó számára új autó regisztrációját kezeli. A rendszer először ellenőrzi, hogy a felhasználónak még nincs regisztrált autója, illetve, hogy minden szükséges adat (plate, height, type) meg van-e adva. Ezt követően ellenőrzi, hogy a megadott rendszámú autó nem szerepel-e már az adatbázisban, majd létrehozza az új autót, és frissíti a felhasználó rekordját az autó azonosítójával.
    - **Autentikáció:** Érvényes JWT token szükséges.
    - **Paraméterek:**
      * **Request Body-ban várható mezők:**
      * plate – az autó rendszáma;
      * height – az autó magassága;
      * type – az autó típusa.
    - **Válasz:** Sikeres regisztráció esetén a rendszer 201-es státusszal az "Az autó sikeresen regisztrálva lett!" üzenetet és a regisztrált autó adatait küldi vissza.
    - **Hibakezelés:**
      * Ha a bejelentkezett felhasználó nem található, 404-es hibakódot és "Felhasználó nem található!" hibaüzenetet küld vissza.
      * Ha a felhasználónak már van regisztrált autója, 400-as hibakódot és "A felhasználónak már van regisztrált autója!" hibaüzenetet küld vissza.
      * Hiányzó adatok esetén 400-as hibakódot és "Hiányzó adat az autó regisztrációhoz!" hibaüzenetet küld vissza.
      * Ha a rendszám már regisztrálva van, 400-as hibakódot és "Ez a rendszám már regisztrálva van!" hibaüzenetet küld vissza.
      * Autó regisztrációs hiba esetén 500-as hibakódot és "Hiba történt az autó regisztrációja során!" hibaüzenetet küld vissza.
  + Saját autó adatainak lekérdezése:
    - **Végpont:** /myCar
    - **Metódus:** GET
    - **Leírás:** A bejelentkezett felhasználóhoz tartozó regisztrált autó adatait kérdezi le a rendszer.
    - **Autentikáció:** Érvényes JWT token szükséges.
    - **Paraméterek:**
      * A felhasználó azonosítója a tokenből kerül meghatározásra; nem vár további paramétereket.
    - **Válasz:** Sikeres lekérdezés esetén a rendszer visszaadja a felhasználóhoz tartozó autó adatait.
    - **Hibakezelés:**
      * Ha a felhasználóhoz nem tartozik regisztrált autó, 404-es hibakódot és "Nincs regisztrált autó ehhez a felhasználóhoz." üzenetet küld vissza.
      * Lekérdezési hiba esetén 500-as hibakódot és "Hiba történt az autó adatok lekérésekor." hibaüzenetet küld vissza.
  + Saját autó adatainak frissítése:
    - **Végpont:** /updateCar
    - **Metódus:** PUT
    - **Leírás:** A bejelentkezett felhasználóhoz tartozó regisztrált autó adatainak módosítása. A rendszer ellenőrzi, hogy legalább egy módosítandó adat (plate, height, type) meg van-e adva, majd a meglévő értékekhez képest frissíti az autó adatokat.
    - **Autentikáció:** Érvényes JWT token szükséges.
    - **Paraméterek:**
      * **Request Body-ban várható mezők:**
        + plate – új rendszám;
        + height – új magasság;
        + type – új típus.
    - **Válasz:** Sikeres módosítás esetén a rendszer az "Az autó adatai sikeresen frissítve!" üzenetet és a frissített autó adatait küldi vissza.
    - **Hibakezelés:**
      * Ha egyetlen módosítandó adat sincs megadva, 400-as hibakódot és "Nincs megadva módosítandó adat!" üzenetet küld vissza.
      * Ha az autó nem található vagy nem tartozik a bejelentkezett felhasználóhoz, 404-es hibakódot és "Az autó nem található vagy nem tartozik hozzád!" hibaüzenetet küld vissza.
      * Módosítási hiba esetén 500-as hibakódot és "Hiba történt az autó adatainak módosításakor!" hibaüzenetet küld vissza.
  + Parkolóházak lekérdezése:
    - **Végpont:** /parkhouses
    - **Metódus:** GET
    - **Leírás:** A rendszer lekéri az összes parkolóház adatait az adatbázisból.
    - **Autentikáció:** Nem szükséges, ez az végpont nyilvános.
    - **Válasz:** Sikeres lekérdezés esetén a rendszer visszaadja az összes parkolóház adatait tartalmazó tömböt.
    - **Hibakezelés:**
      * Ha hiba történik a lekérdezés során, 500-as hibakódot és "Nem sikerült lekérni a parkolóházakat" hibaüzenetet küld vissza.
  + Jelszó megváltoztatása:
    - **Végpont:** /changePassword
    - **Metódus:** PUT
    - **Leírás:** A bejelentkezett felhasználó a régi jelszavának ellenőrzése után megadhatja az új jelszót. A rendszer a régi jelszó validálása után hash-eli az új jelszót, majd frissíti a felhasználó adatait az adatbázisban.
    - **Autentikáció:** Érvényes JWT token szükséges.
    - **Paraméterek:**
      * **Request Body-ban várható mezők:**
        + oldPassword – a jelenlegi jelszó;
        + newPassword – az új jelszó.
    - **Válasz:** Sikeres jelszó módosítás esetén az "Jelszó sikeresen frissítve!" üzenet jelenik meg.
    - **Hibakezelés:**
      * Ha a felhasználó nem található, 404-es hibakódot és "Felhasználó nem található!" hibaüzenetet küld vissza.
      * Ha a régi jelszó érvénytelen, 401-es hibakódot és "Hibás régi jelszó!" hibaüzenetet küld vissza.
      * Jelszó módosítási hiba esetén 500-as hibakódot és "Hiba történt a jelszó módosítása során!" hibaüzenetet küld vissza.
  + Foglalás hozzáadása:
    - **Végpont:** /reserve
    - **Metódus:** POST
    - **Leírás:** A rendszer a bejelentkezett felhasználó számára új foglalás létrehozását teszi lehetővé; ehhez kötelezően meg kell adni a parkolóhely sorszámát (park\_slot), az adott parkolóház azonosítóját (parkhouse\_id), a foglalás időtartamát órában (reservation\_time\_hour) és a foglalás indulási idejét (start\_time); a foglalás tulajdonosának azonosítója a tokenből kerül kinyerésre.
    - **Autentikáció:** Érvényes JWT token szükséges az authenticate() middleware miatt.
    - **Paraméterek:**
      * **Request Body-ban várható mezők:**
        + park\_slot – a foglalni kívánt parkolóhely sorszáma;
        + parkhouse\_id – az adott parkolóház azonosítója;
        + reservation\_time\_hour – a foglalás időtartama órában;
        + start\_time – a foglalás kezdési ideje.
      * A foglalást végző felhasználó azonosítója a tokenből kerül meghatározásra (reservation\_owner\_id).
    - **Válasz:** Sikeres foglalás létrehozása esetén 201-es státuszkóddal a rendszer az „Foglalás sikeresen létrehozva!” üzenetet küldi vissza, valamint a létrehozott foglalási adatokat tartalmazó objektumot.
    - **Hibakezelés:**
      * Ha valamely kötelező adat hiányzik, 400-as hibakóddal és „Hiányzó adatok!” hibaüzenettel küld vissza;
      * Ha a felhasználónak már van aktív foglalása, 400-as hibakóddal és „Már rendelkezel aktív foglalással!” hibaüzenettel küld vissza;
      * Ha az adott parkolóhely már foglalt, 400-as hibakóddal és „A parkolóhely már foglalt!” hibaüzenettel küld vissza;
      * Foglalási hiba esetén 500-as hibakóddal és „Hiba történt a foglalás során!” hibaüzenettel válaszol.
  + Foglalt helyek lekérdezése:
    - **Végpont:** /parkhouse/reservedSlots/:id
    - **Metódus:** GET
    - **Leírás:** A rendszer az adott parkolóházhoz tartozó, aktív foglalások alapján kinyeri a foglalt parkolóhelyek sorszámait; a parkolóház azonosítója (id) az URL-ben kerül megadásra.
    - **Paraméterek:**
      * Útvonalparaméter:
        + id – az adott parkolóház azonosítója.
    - **Válasz:** Sikeres lekérdezés esetén 200-as státuszkóddal a rendszer a foglalt parkolóhelyek sorszámát tartalmazó tömböt küldi vissza az „reservedSlots” kulcs alatt.
    - **Hibakezelés:**
      * Lekérdezési hiba esetén 500-as hibakóddal és „Hiba történt a foglalt helyek lekérdezésekor!” hibaüzenettel válaszol.
  + Előző és jelenlegi foglalások lekérdezése:
    - **Végpont:** /reservations/my
    - **Metódus:** GET
    - **Leírás:** A rendszer a bejelentkezett felhasználó tokenéből kinyert azonosító alapján lekéri az adott felhasználó összes foglalását; a foglalásokat aktív (active = true) és inaktív (active = false vagy inactive = true) csoportokba válogatja.
    - **Autentikáció:** Érvényes JWT token szükséges.
    - **Paraméterek:**
      * A felhasználó azonosítója a tokenből kerül meghatározásra; nem vár további URL, query vagy body paramétereket.
    - **Válasz:** Sikeres lekérdezés esetén 200-as státuszkóddal a rendszer az aktív foglalásokat az „activeReservations”, az inaktív foglalásokat pedig az „inactiveReservations” kulcs alatt küldi vissza.
    - **Hibakezelés:**
      * Lekérdezési hiba esetén 500-as hibakóddal és „Hiba történt a foglalások lekérésekor!” hibaüzenettel válaszol.

# Github

[**Github link**](https://github.com/VidekiCorleone/FinalProject)

# Források

Tartalomjegyzék

[I. Feladatspecifikáció 3](#_Toc197953837)

[II. Alkalmazott szoftverek és technológiák 4](#_Toc197953838)

[III. A felület funkciói, felépítése és használata 5](#_Toc197953839)

[IV. Adatbázis 29](#_Toc197953840)

[V. API végpontok 34](#_Toc197953841)

[VI. Github 45](#_Toc197953842)

[VII. Források 45](#_Toc197953843)