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.

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.

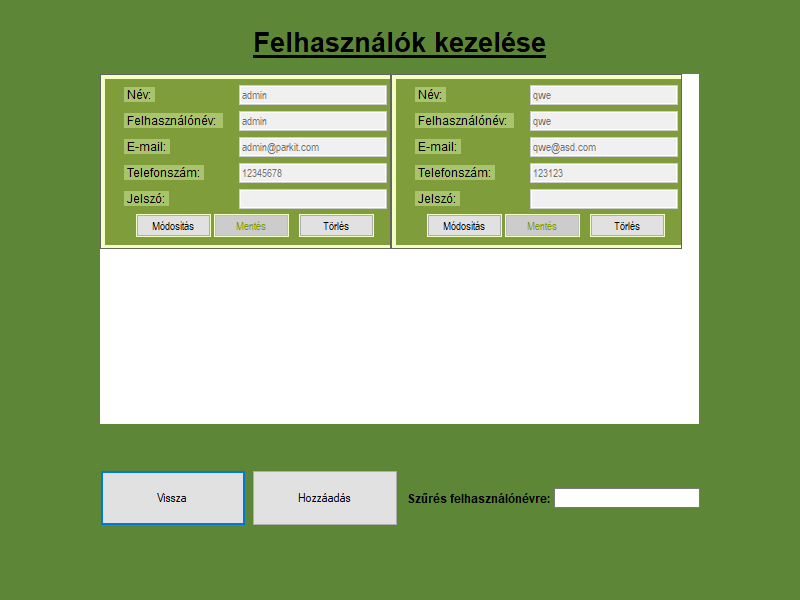
Abban az esetben, ha a bejelentkezés sikeres, a fő menü felülete megjelenik. A fő menüben 3 további választási opciót látunk. felhasználók kezelése, Parkolóházak kezelése és Foglalások kezelése, illetve egy kijelentkezés gombot.

****

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

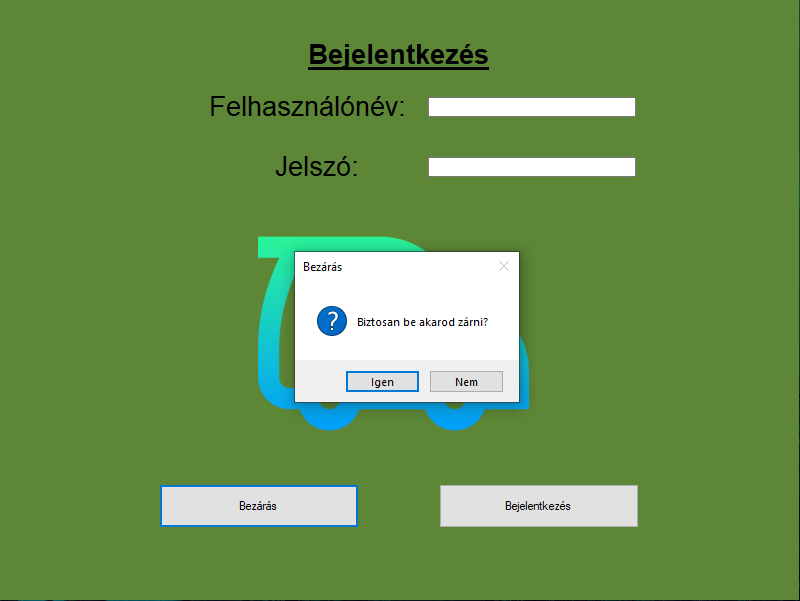
Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

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**
  2. **Reszponzív weboldal**

# Adatbázis

# 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 várható 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.

# Tesztelési dokumentáció

# Források

Tartalomjegyzék

[I. Feladatspecifikáció 3](#_Toc197809236)

[II. Alkalmazott szoftverek és technológiák 4](#_Toc197809237)

[III. A felület funkciói, felépítése és használata 5](#_Toc197809238)

[IV. Adatbázis 17](#_Toc197809239)

[V. API végpontok 17](#_Toc197809240)

[VI. Tesztelési dokumentáció 22](#_Toc197809241)

[VII. Források 22](#_Toc197809242)