Rentaruro

Autókölcsönző

Készítette:

Kántor Krisztián Péter

Sebestyén Milán Gábor

Városi Bendegúz Álmos

Nyilatkozat

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

# Feladatspecifikáció

# Alkalmazott szoftverek és technológiák

# Projektmunka talapítése

# A felület funkciói, felépítése és használata

# Adatbázis

# API végpontok

# Tesztelési dokumentáció

# Források

Tartalomjegyzék

[I. Feladatspecifikáció 5](#_Toc197517653)

[II. Alkalmazott szoftverek és technológiák 6](#_Toc197517654)

[III. Projektmunka telepítése 8](#_Toc197517655)

[IV. A felület funkciói, felépítése és használata 9](#_Toc197517656)

[V. Adatbázis 13](#_Toc197517657)

# Feladatspecifikáció

Az általunk fejlesztett alkalmazás segíti egy autókölcsönző mindennapjait, a felhasználónak egy egyszerűbb és könnyebben használható felületet biztosít a jobb élményért.

A Rentaruro több felhasználói szinttel rendelkezik: ügyfél, dolgozó és admin. A célja az autófoglalás illetve az adminisztratív feladatok hatékony és precíz elvégzése.

Célunk hogy minél több embernek nyújtsunk elérhető áron autókat, azok naprakészségét, jól felszereltségét biztosítsuk, hogy minden ember zökkenőmentesen és biztonságosan járják az utakat.

# Alkalmazott szoftverek és technológiák

* 1. Szoftverek:
* XAMPP:

phpMyAdmin 5.2.1

MySQL 10.3.38

Apache 2.4.56

MariaDB 10.4.28

* Google Chrome Verzió: 123.0.6312.86
* Microsoft Word 2021 LTSC
* Adobe Photoshop 2023
* GitHub Desktop v3.3.8
* Visual Studio Code 1.87.2
* Android Studio 2024.3.1
* Visual Studio Community 2022 17.12.4
* IntelliJ IDEA 2025.1
  1. Programnyelvek, keretrendszerek és modulok

**Frontend**

* Nyelv: JavaScript
* Keretrendszer: React.js
* UI komponenskönyvtár:
* Routing: React Router DOM
* React: 18.3.1
* React Query (TanStack Query): 5.56.2
* Vite
* Lucide React
* Tailwind CSS
* Shadcn/UI

**Backend**

* Nyelv: JavaScript (Node.js)
* Web Framework: Express.js
* Adatbázis Kezelő: MySQL
* Authentikáció és Autorizáció: bcrypt, jsonwebtoken
* HTTP Kliens: axios
* CSV Fájlkezelő: csv-parser
* Környezeti változók kezelése: dotenv
* Middleware: body-parser, cors, multer
* Egyéb: assert(beépített Node.js modul)

**Tesztelés**: ???????

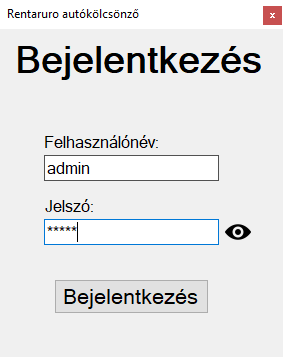
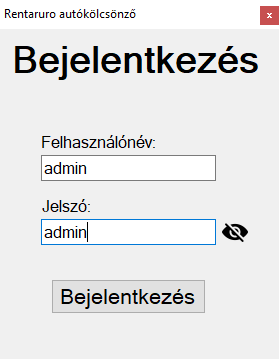
# Projektmunka telepítése

# A felület funkciói, felépítése és használata

A felületek felépítésénél számos dologra törekedtünk, de legfőképp az egyszerűségre és az átláthatóságra.

1. Natív asztali alkalmazás

Ezen a felületen lehet kezelni az autókat, kölcsönzéseket és a dolgozókat. Ezekhez azonban csak belépéssel lehet hozzáférni, ezért a szerver létrehoz egy új felhasználót admin névvel és admin jelszóval.

* 1. Autók

Bejelentkezés után megjelenik egy új ablak, ahol egy listában egymás alatt láthatók az autók adatai: rendszám, márka, model, évjárat, meghajtás, váltó, üzemanyag, klíma, tolatóradar, tempomat, árak, kaució, kategória. Ezek az adatok alapján lehet csökkenő, vagy növekvő sorrndbe állítani. Autót hozzáadni, törölni és módosítani kizárólag admin jogosultsággal rendelkező felhasználó képes. Egy rendszám csak egyszer szerepelhet az adatbázisban.

Hozzáadáshoz szükséges adatok: rendszám, márka, model, fénykép az autóról, évjárat, meghajtás, váltó, üzemanyag, klíma, tolatóradar, tempomat, ár 1 és 5 nap között (Ft/nap), ár 6 és 14 nap között (Ft/nap), ár 15 nap fölött (Ft/nap), kaució, kategória, egyéb információ az autóról. A hozzáadás gombra kattintva hozzáadja az autót az adatbázishoz, a mégse gombra kattintva üresre állítja az adatokat. Minden adatot ki kell tölteni, hogy sikeresen hozzáadja az autót az adatbázishoz.

Az elemek jobb oldalán látható 🔧-gombra kattintva visszatölti az autó adatait és a Hozzáadás gomb szövegét átírja Módosítás-ra. A gomb lenyomásával frissíti az autó adatait. Módosítani a rendszámon kívül minden adatot lehet.

A 🔧 -gomb helyett dolgozó hozzádérésel rendlekezőknek ⓘ -gomb jelenik meg. A gomb lenyomásával megtekinthetik az autó további információit, amelyet a lista nem tartalmaz.

A ✖ -gombra kattintva megjelenik egy ablak, amely megkérdezi, hogy biztosan szeretné törölni az autót az adatbázisból. Az igen gombra kattintva az autót kitörli.

* 1. Kölcsönzések

A Kölcsönzések gombra kattintva megjelenik a kölcsönzések listája. Ez a lista a kölcsönzés kezdetét, végét az autó rendszámát, márkáját, modeljét a felhasználó nevét, emailjét, telefonszámát, felhasználónevét és egyébb kérését tartalmazza. Az elemek egymás alatt jelennek meg. Új kölcsönzést ezen a felületen nem lehet létrehozni, mindössze a meglévők hosszát és egyébb információit lehet módosítani, vagy a kölcsönzést törölni.

Az elemek jobb oldalán látható 🔧-gombra kattintva visszatölti a bérlés kezdetét, végét és egyébb információját. Ha ugyan arra az autóra van több foglalás akkor a foglalást nem lehet úgy módosítani, hogy azok időpontjai bármilyen módon egybeessenek. A módosítás gomb lenyomásával frissíti az autó adatait.

A ✖ -gombra kattintva megjelenik egy ablak, amely megkérdezi, hogy biztosan szeretné törölni a foglalást az adatbázisból. Az igen gombra kattintva törli a foglalást.

* 1. Dolgozók

A Dolgozók gomb kizárólag admin jogosultságga rendelkező felhasználóknak jelenik meg. A gombra kattintva egy listában látható az összes dolgozó felhasználóneve, emailje, telefonszáma, szöletési dátuma, adószáma, lakcíme, neve és beosztása.

Új dolgozó létrehozásához szükséges adatok: felhasználónév, jelszó, email, telefonszám, adószám, születési dátum, lakcím, néve, beosztás (admin/dolgozó). Két ugyanolyan felhasználónévvel rendelkező dolgozót nem lehet létrehozni.

Az elemek jobb oldalán látható 🔧-gombra kattintva visszatölti a felasználó adatait és a Hozzáadás gomb szövegét megváltoztatja Módosítás-ra. A gombra kattintva frissíti a dolgozó adatait. Minden adatot meg lehet változtatni

A ✖ -gombra kattintva megjelenik egy ablak, amely megkérdezi, hogy biztosan szeretné törölni az adott felhasználót. Az igen gombra kattintva törli a felhasználót.

A jobb felső sarokban található Kijelentkezés gombra kattintva bezárja az ablakot és újra megnyitja a bejelentkezési felületet.

1. Kliens oldali komponens

Ezen a felületen lehet felhasználót létrehozni, létrehozott felhasználóba bejelentkezni, új foglalást létrehozni. Foglalni csak akkor lehet, ha már be van jelentkezve a felhasználó.

* 1. Felhasználó létrehozása

Új felhasználóhoz szükséges adatok: felhasználónév, jelszó, telefonszám, emali, név.

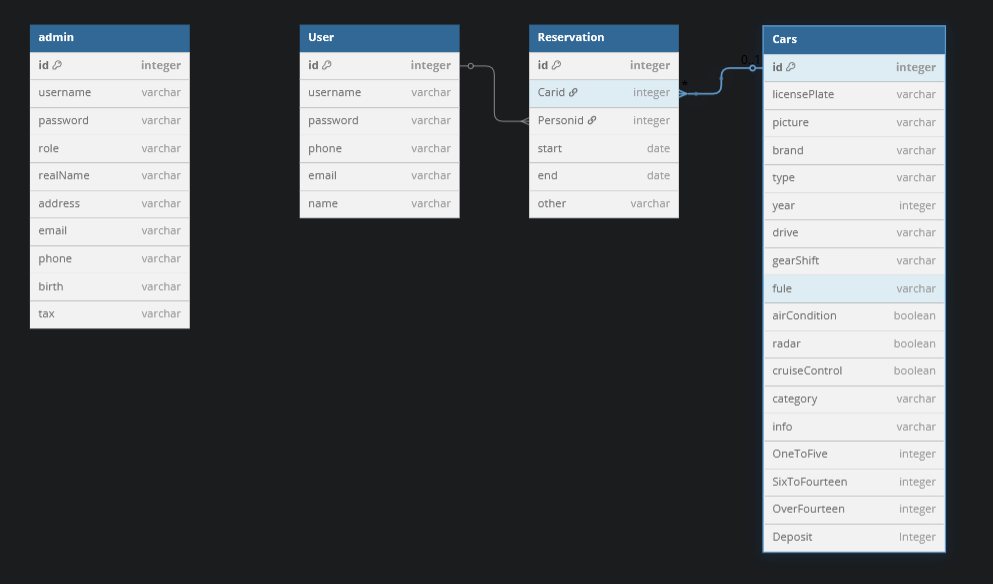
1. Natív mobil alkalmazás

Ezen a felületen lehet felhasználót létrehozni, létrehozott felhasználóba bejelentkezni, új foglalást létrehozni. Foglalni csak akkor lehet, ha már be van jelentkezve a felhasználó.

* 1. Felhasználó létrehozása

Új felhasználóhoz szükséges adatok: felhasználónév, jelszó, telefonszám, emali, név.

# Adatbázis



1. API végpontok

Autók:

1. Autó hozzáadás

* Végpont: /AddCar
* Metódus: POST
* Leírás:
  + Ez a végpont létrehoz egy új autót
* Paraméter:
  + Fénykép, rendszám, márka, model, évjárat, meghajtás, váltó, üzemanyag, tolatóradar, tempomat, egyéb információ, kategória, 1-5 nap (Ft/nap), 6-14 (Ft/nap), 15- (Ft/nap), kaució, autentikáció
* Hibakezelés:
  + Ha nincs token akkor egy üzenetet "Nem létező token!" és 401-es (unauthorized) hibaüzenetet küld vissza
  + Ha rossz a token akkor "Hibás token!" üzenetet és 498-es hibakódot (invalid token) küld vissza
  + Ha már létezik ezzel a rendszámmal autó, akkor 409-es hibakódot (conflict) és azt küldi vissza, hogy "Már létezik ezzel a rendszámmal autó!"

1. Autók lekérdezése

* Végpont: /ListCars
* Metódus: GET
* Leírás:
  + Ez a végpont lekérdezi az összes autót
* Hibakezelés:
  + Sikertelen lekérdezés esetén 400-as hibakódot (bad request) küld vissza

1. Autó törlése

* Végpont: /DeleteCar/:id
* Metódus: DELETE
* Leírás:
  + Ez a végpont törli a paraméterkén megadott id-val rendelkező autót
* Paraméter:
  + id: ez alapján találja meg a program melyik autót törölje, autentikáció
* Hibakezelés:
  + Ha nincs token akkor egy üzenetet "Nem létező token!" és 401-es hibaüzenetet (unauthorized) küld vissza
  + Ha rossz a token akkor "Hibás token!" üzenetet és 498-es hibakódot (invalid token) küld vissza
  + Ha sikertelen a lekérdezés 400-as hibakódot (bad request) ad vissza
  + Ha nem találja az autót "Nem létezik ezzel az id-val autó!" üzenetet és 404-es errort (not found) küld

1. Aútó módosítása

* Végpont: /UpdateCar/:id
* Metódus: PUT
* Leírás:
  + Ez a végpont tudja módosítani az autók adatait, kivéve a rendszámot
* Paraméter:
  + id: ez alapján tudja módosítani a program az autó adatait, autentikáció
* Hibakezelés:
  + Ha nincs token akkor egy üzenetet "Nem létező token!" és 401-es hibaüzenetet (unauthorized) küld vissza
  + Ha rossz a token akkor "Hibás token!" üzenetet és 498-es hibakódot (invalid token) küld vissza
  + Ha nem találja az autót 404-es errort (not found) és "Nincs ilyen autó!"üzenetet küld vissza
  + Ha sikertelen a lekérdezés 400-as hibakódot (bad request) ad vissza

Kölcsönzések:

1. Új kölcsönzés

* Végpont: /NewRent
* Metódus: POST
* Leírás:
  + Ez a végponttal lehet új kölcsönzést létrehozni
* Paraméter:
  + carId: a kiválasztott autó id-je, personId: a felhasználó id-je, kezdete, vége, egyébb kérés, autentikáció
* Hibakezelés:
  + Ha nincs token akkor egy üzenetet "Nem létező token!" és 401-es hibaüzenetet (unauthorized) küld vissza
  + Ha rossz a token akkor "Hibás token!" üzenetet és 498-es hibakódot (invalid token) küld vissza

1. Kölcsönzések lekérdezése

* Végpont: /ListAllRents
* Metódus: GET
* Leírás:
  + Ez a végpont lekérdezi az összes kölcsönzést (id, kezdete, vége, egyébb kérés, autó id, rendszám, márka, model, felhasználó id, felhasználónév, név, email, telefonszám)
* Paraméter:
  + autentikáció
* Hibakezelés:
  + Ha nincs token akkor egy üzenetet "Nem létező token!" és 401-es hibaüzenetet (unauthorized) küld vissza
  + Ha rossz a token akkor "Hibás token!" üzenetet és 498-es hibakódot (invalid token) küld vissza
  + Sikertelen lekérdezésnél 400-as error kódot (bad request) küld vissza

1. Kölcsönzés törlése

* Végpont: /DeleteRent/:id
* Metódus: DELETE
* Leírás:
  + Ezzel a végponttal lehet törölni a kölcsönzést
* Paraméter:
  + id: ez alapján keresi meg a kölcsönzést, autentikáció
* Hibakezelés:
  + Ha nincs token akkor egy üzenetet "Nem létező token!" és 401-es hibaüzenetet (unauthorized) küld vissza
  + Ha rossz a token akkor "Hibás token!" üzenetet és 498-es hibakódot (invalid token) küld vissza
  + Sikertelen lekérdezésnél 400-as error kódot (bad request) küld vissza
  + Ha nem találja a foglalást akkor 404-es hibakódot (not found) és "Nem található ilyen foglalás!" üzenetet küld vissza

1. Kölcsönzés módosítása

* Végpont: /DeleteRent/:id
* Metódus: PUT
* Leírás:
  + Ezzel a végponttal lehet a kölcsönzések kezdetét, végét és egyébb kérést módosítani
* Paraméter:
  + id: ez alapján keresi meg a kölcsönzést, autentikáció
* Hibakezelés:
  + Ha nincs token akkor egy üzenetet "Nem létező token!" és 401-es hibaüzenetet (unauthorized) küld vissza
  + Ha rossz a token akkor "Hibás token!" üzenetet és 498-es hibakódot (invalid token) küld vissza
  + Sikertelen lekérdezésnél 400-as error kódot (bad request) küld vissza
  + Ha nem találja a foglalást akkor 404-es hibakódot (not found) és "Nem található ilyen foglalás!" üzenetet küld vissza

Dolgozók:

1. Regisztráció

* Végpont: /AdminRegistration
* Metódus: POST
* Leírás:
  + Ez a végpont hoz létre új dolgozói felhasználót
* Paraméter:
  + felhasználónév, jelszó, beosztás, név, lakcím, email, telefonszám, születési dátum, adószám, autentikáció
* Hibakezelés:
  + Ha nincs token akkor egy üzenetet "Nem létező token!" és 401-es hibaüzenetet (unauthorized) küld vissza
  + Ha rossz a token akkor "Hibás token!" üzenetet és 498-es hibakódot (invalid token) küld vissza
  + Sikertelen lekérdezésnél 400-as error kódot (bad request) küld vissza
  + Ha már létezik ezzel a felhasználónévvel fiók akkor 409-es errort (conflict) és 'Már létezik ilyen felhasználó!' üzenetet küld vissza

1. Bejelentkezés

* Végpont: /AdminLogin
* Metódus: POST
* Leírás:
  + Ezzel a végponttal lehet bejelentkezni a dolgozói felületre
* Paraméter:
  + felhasználónév, jelszó
* Hibakezelés:
  + Sikertelen lekérdezésnél 400-as error kódot (bad request) küld vissza
  + Ha rossz a felhasználónév vagy jelszó akkor 409-es error kódot (conflict) és "Hibás felhasználónév, vagy jelszó" üzenetet küld vissza

1. Dolgozók kilistázása

* Végpont: /ListAllWorkers
* Metódus: GET
* Leírás:
  + Ezzel a végponttal lehet kilistázni a dolgozókat
* Paraméter:
  + autentikáció
* Hibakezelés:
  + Sikertelen lekérdezésnél 400-as error kódot (bad request) küld vissza

1. Dolgozó törlése

* Végpont: /DeleteWorker/:id
* Metódus: DELETE
* Leírás:
  + Ezzel a végponttal lehet törlölni egy dolgozó felhasználóját
* Paraméter:
  + id: ez alapján keresi meg a dolgozót, autentikáció
* Hibakezelés:
  + Ha nincs token akkor egy üzenetet "Nem létező token!" és 401-es hibaüzenetet (unauthorized) küld vissza
  + Ha rossz a token akkor "Hibás token!" üzenetet és 498-es hibakódot (invalid token) küld vissza
  + Sikertelen lekérdezésnél 400-as error kódot (bad request) küld vissza
  + Ha nem találja a dolgozót akkor 404-es error kódot (not found) és "Nincs ilyen dolgozó!" üzenetet küld vissza

1. Dolgozó módosítása

* Végpont: /UpdateWorker/:id
* Metódus: PUT
* Leírás:
  + Ezzel a végponttal lehet a dolgozók adatait módosítani
* Paraméter:
  + id: ez alapján keresi meg a dolgozót, autentikáció
* Hibakezelés:
  + Ha nincs token akkor egy üzenetet "Nem létező token!" és 401-es hibaüzenetet (unauthorized) küld vissza
  + Ha rossz a token akkor "Hibás token!" üzenetet és 498-es hibakódot (invalid token) küld vissza
  + Sikertelen lekérdezésnél 400-as error kódot (bad request) küld vissza
  + Ha nem találja a dolgozót akkor 404-es error kódot (not found) és "Nincs ilyen dolgozó!" üzenetet küld vissza
  + Ha már létezik ilyen felhasználónév akkor 409-es errorkódot (conflict) "Már létezik ilyen felhasználó!" üzenettel ad vissza