**Vizsgaremek Projekt Dokumentáció**

**Projekt címe: *Car Rental app* – Webes és mobilalkalmazás**

**1. A szoftver célja**

A projekt célja egy modern, felhasználóbarát autóbérlő alkalmazás létrehozása, amely lehetővé teszi a felhasználók számára, hogy különféle járműveket béreljenek webes vagy mobilfelületen keresztül. A rendszer adminisztrátori és felhasználói szerepköröket különít el, támogatja a foglalások, biztosítások és extrák kezelését, valamint rendelkezik bejelentkezési és jogosultságkezelési funkciókkal is.

**2. Főbb Funkciók**

* Autók böngészése és részletes információk megtekintése
* Foglalási rendszer időpont választással
* Felhasználói fiók kezelése
* Adminisztrátori felület a járművek és foglalások kezeléséhez
* Biztosítási és extra szolgáltatások választása
* Helyszín és elérhetőség megjelenítése

**3. A rendszer technikai leírása**

**Frontend:**

* Next.js (React)
* Tailwind CSS
* Framer Motion (animációkhoz)
* TypeScript

**Backend:**

* NestJS
* Prisma (ORM)
* MySQL adatbázis
* Swagger API dokumentáció (localhost:3000/apidoc)
* Token alapú autentikáció
* Argon2 jelszó titkosítás

**Mobilalkalmazás:**

* Flutter (Android/iOS támogatással)

**4. Működési feltételek**

* Node.js és npm telepítve a backendhez és frontendhez
* MySQL szerver a relációs adatbázishoz
* React környezet a frontend fejlesztéséhez
* Flutter SDK a mobilalkalmazás fejlesztéséhez

**5. Futtatási Környezet**

* Backend: 3000-es port
* Frontend: 8080-as port
* MySQL adatbázis: default port (3306)

**6. Adatbázismodell**

**Az adatbázis neve: rentcar**

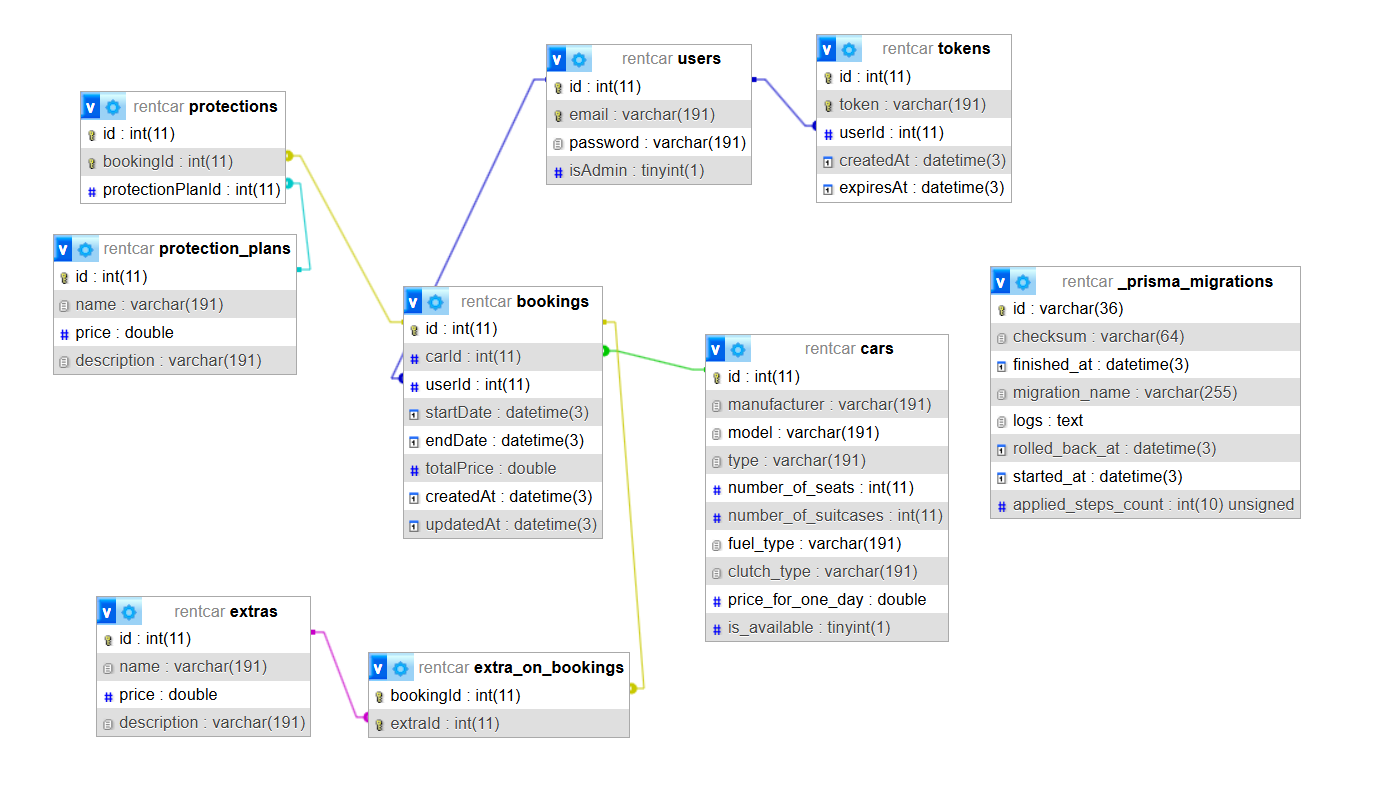
**Főbb Entitások:**

* Car (Autók)
* User (Felhasználók)
* Booking (Foglalások)
* Protection (Biztosítások)
* Extra (Kiegészítő szolgáltatások)
* Token (Autentikációs tokenek)

**Kapcsolatok:**

* Egy autóhoz több foglalás tartozhat
* Egy felhasználóhoz több foglalás tartozhat
* Minden foglaláshoz egy biztosítási csomag választható
* Foglalásokhoz több extra szolgáltatás is tartozhat

Az adatbázismodell Prisma ORM segítségével került megvalósításra. Az entitások közti kapcsolatokat relációk írják le (pl. egy foglalás több extrát is tartalmazhat).



**7. Swagger dokumentáció / API Végpontok**

**Elérheto:** <http://localhost:3000/apidoc>

**Autentikáció:**

* POST /auth/login - Bejelentkezés
* POST /auth/register - Regisztráció
* GET /auth/profile - Bejelentkezett user adatai
* DELETE /auth/logout - Kijelentkezes

**Autók Kezelése:**

* GET /cars - Autók listázása
* GET /cars/:id - Egy autó részletes adatai
* POST /cars - Új autó hozzáadása (admin)
* PATCH /cars/:id – Egy autó módosítása (admin)
* DELETE /cars/:id – Egy autó törlése (admin)

**Foglalások:**

* GET /bookings - Foglalások listázása
* POST /bookings - Új foglalás létrehozása
* GET /bookings/:id – Egy foglalás részletei
* PATCH /bookings/:id – Egy foglalás módosítása
* DELETE /bookings/:id – Egy foglalás törlése
* GET /bookings/extras - Kiegészítők listája
* GET /bookings/protections - Biztositások listája

**Felhasználók:**

* GET /users - Felhasználók listázása (admin)
* POST /users - Felhasználó létrehozása
* GET /users/:id – Egy felhasználó adatai
* PATCH /users/:id – Egy felhasználó módosítása
* DELETE /users/:id – Egy felhasználó törlése (admin)

**DTO:**

* CreateUserDto
* UpdateUserDto
* LoginDto
* CreateCarDto
* UpdateCarDto
* CreateBookingDto
* UpdateBookingDto

**Biztonsági Megoldások:**

* Autentikáció
  + Argon2 jelszó hashelés
  + Bearer token autorizáció
* Jogosultságkezelés
  + Admin és felhasználói szerepkörök
  + Védett végpontok AuthGuard használatával
  + Token validáció minden védett műveletnél



**8. Tesztkódok és eredmények**

A backend API végpontjai Flashpost/Swagger segítségével lettek tesztelve (exportfájl):

* Bejelentkezés, regisztráció
* Felhasználó létrehozása, lekérdezése
* Autó lekérdezése, létrehozása
* Foglalás létrehozása, lekérdezése

Az összes végpont a várt eredményt adta vissza.

Sikeres válasz: HTTP 200, hiba esetén: megfelelő HTTP hibakód (pl. 400, 401).