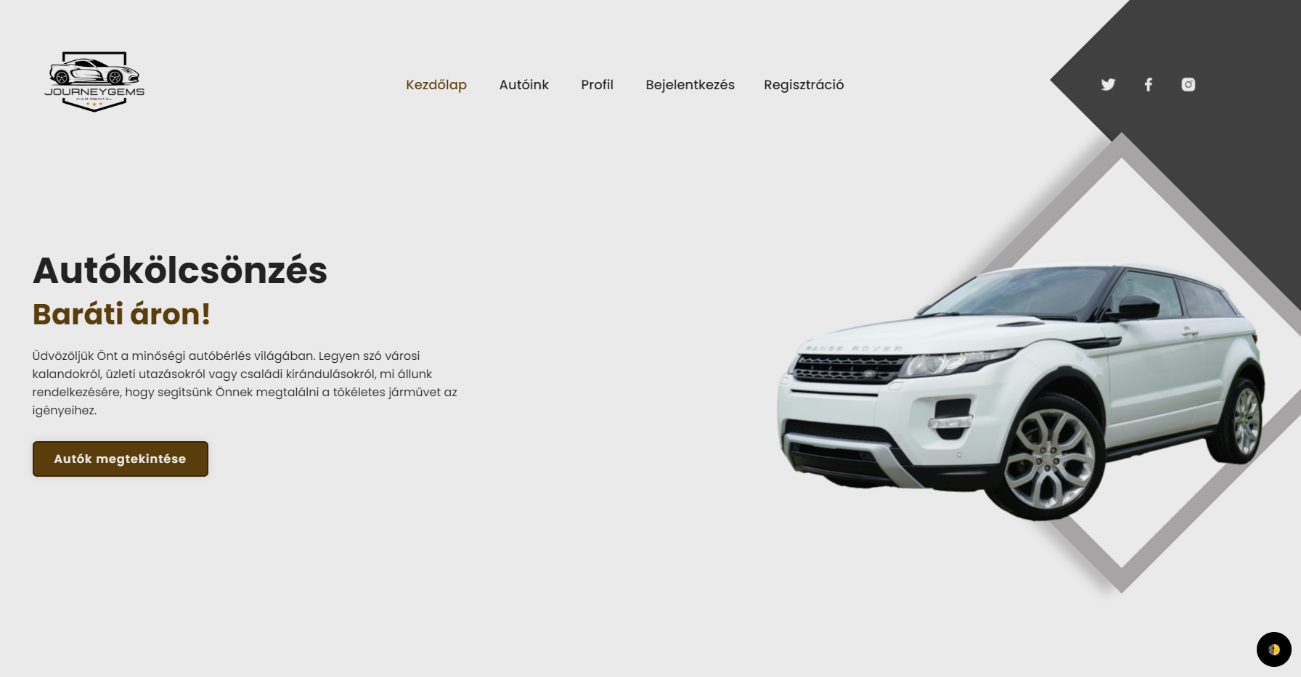
2024

Beke Szabolcs; Szlobodnyik Milán

2/14. SZN

2024.04.12.

JourneyGems



Autóbérlős weboldal

Tartalom

[Bevezetés 3](#_Toc164241717)

[1. Felhasználói dokumentáció 4](#_Toc164241718)

[1.1. Kezdő oldal 4](#_Toc164241719)

[1.2. A menü 5](#_Toc164241720)

[1.3. Bérelhető autóink oldala 6](#_Toc164241721)

[1.4. Foglalós oldala a kiválasztott autónak 8](#_Toc164241722)

[1.5. Profil Oldal 11](#_Toc164241723)

[1.6. Bejelentkezés 12](#_Toc164241724)

[1.7. Regisztráció 13](#_Toc164241725)

[2. Fejlesztői dokumentáció 14](#_Toc164241726)

[2.1. Felhasznált technológiák 14](#_Toc164241727)

[2.1.1. VSC 14](#_Toc164241728)

[2.1.2. HTML 14](#_Toc164241729)

[2.1.3. CSS 14](#_Toc164241730)

[2.1.4. PHP 15](#_Toc164241731)

[2.1.5. JavaScript 15](#_Toc164241732)

[2.1.6. .NET C# 15](#_Toc164241733)

[2.1.7. XAMPP 15](#_Toc164241734)

[2.1.8. Microsoft Word 16](#_Toc164241735)

[2.1.10. Microsoft Excel 16](#_Toc164241736)

[2.1.11. MySQL 16](#_Toc164241737)

[2.1.12. GitHub 16](#_Toc164241738)

[2.1.13. Google Drive 17](#_Toc164241739)

[2.1.14. Discord 17](#_Toc164241740)

[2.1.15. dbdiagram.io 17](#_Toc164241741)

[2.1.16. AngularJS 18](#_Toc164241742)

[2.2. Adatbázis készítésének lépései 18](#_Toc164241743)

[2.2.13. Az adatbázis célja 18](#_Toc164241744)

[2.2.2. Tervezés megkezdése 18](#_Toc164241745)

[2.2.3. Tervezési lépések 18](#_Toc164241746)

[2.2.4. Egyedtípusok/egyedek meghatározása 18](#_Toc164241747)

[2.2.5. Kapcsolatok meghatározása 19](#_Toc164241748)

[2.2.6. N:M kapcsolatok felbontása 20](#_Toc164241749)

[2.2.7. Táblák 20](#_Toc164241750)

[2.2.8. Adatbázis ábrák 24](#_Toc164241751)

[2.2.9. Adatbázis diagram 26](#_Toc164241752)

[2.3. Frontend programozás 27](#_Toc164241753)

[2.3.1. Kezdőoldal 27](#_Toc164241754)

[2.4. Backend programozás 27](#_Toc164241755)

[2.4.1. Adatbázis kapcsolat létrehozása 27](#_Toc164241756)

[2.4.2. Regisztrációs adatok feldolgozása 27](#_Toc164241757)

[2.4.3. Beléptetés 30](#_Toc164241758)

[2.4.4. Profil oldal 31](#_Toc164241759)

[2.3. Tesztelés 34](#_Toc164241760)

[Összefoglalás 36](#_Toc164241761)

[Továbbfejlesztési lehetőségek 37](#_Toc164241762)

[Források: 38](#_Toc164241763)

[Ábrajegyzék 39](#_Toc164241764)

# Bevezetés

A vizsga projektünk témájának kiválasztása nem okozott nehézséget, hiszen mindketten érdeklődünk az autók világa iránt. Ezért úgy döntöttünk, hogy egy olyan weboldalt hozunk létre, ahol a felhasználók a regisztráció után könnyedén tudnak választani az elérhető autók közül, hogy melyiket szeretnék kibérelni.

A felhasználóknak lehetőségük van böngészni a rendelkezésre álló autók között, kiválasztani a számukra megfelelőt, valamint egyszerűen és gyorsan lefoglalni azt a kívánt időszakra. Ezen kívül láthatják, hogy az adott időpontban az autó már le van-e foglalva.

A munkafolyamatokat nem tudjuk élesen ketté választani, hogy melyikünk melyik részét készítette a projektnek, mivel mindketten egyaránt dolgoztunk mind a Frontend, mind a Backend, illetve az adatbázis elkészítését illetően. A projekt során szorosan együttműködtünk és egymást támogattuk a feladatok elvégzésében.

Mindketten számos autókölcsönző weboldalt áttanulmányoztunk, és a legjobb ötleteket és kódrészleteket gyűjtöttük össze annak érdekében, hogy egy valóban üzleties és profi megjelenést biztosítsunk weboldalunknak. Előfordult, hogy az egyikünk összeállította az egyes oldalakat, míg a másikunk kiegészítette saját ötleteivel, így közösen alkottunk egy innovatív és felhasználóbarát felületet.

Dolgozatunk első fejezete a felhasználói dokumentáció, amelyben részletesen bemutatjuk a weboldalunk működését.

A második fejezetben a felhasználói dokumentáció alatt ismertetjük a felhasznált technológiákat, fejlesztői eszközöket, majd ezt követően a programozás azon részleteiről beszélünk, amelyeket fontosnak tartottunk kiemelni.

Végezetül az összefoglalóban a felmerült problémákról és tovább fejlesztési lehetőségekről írtunk.

# Felhasználói dokumentáció

## Kezdő oldal

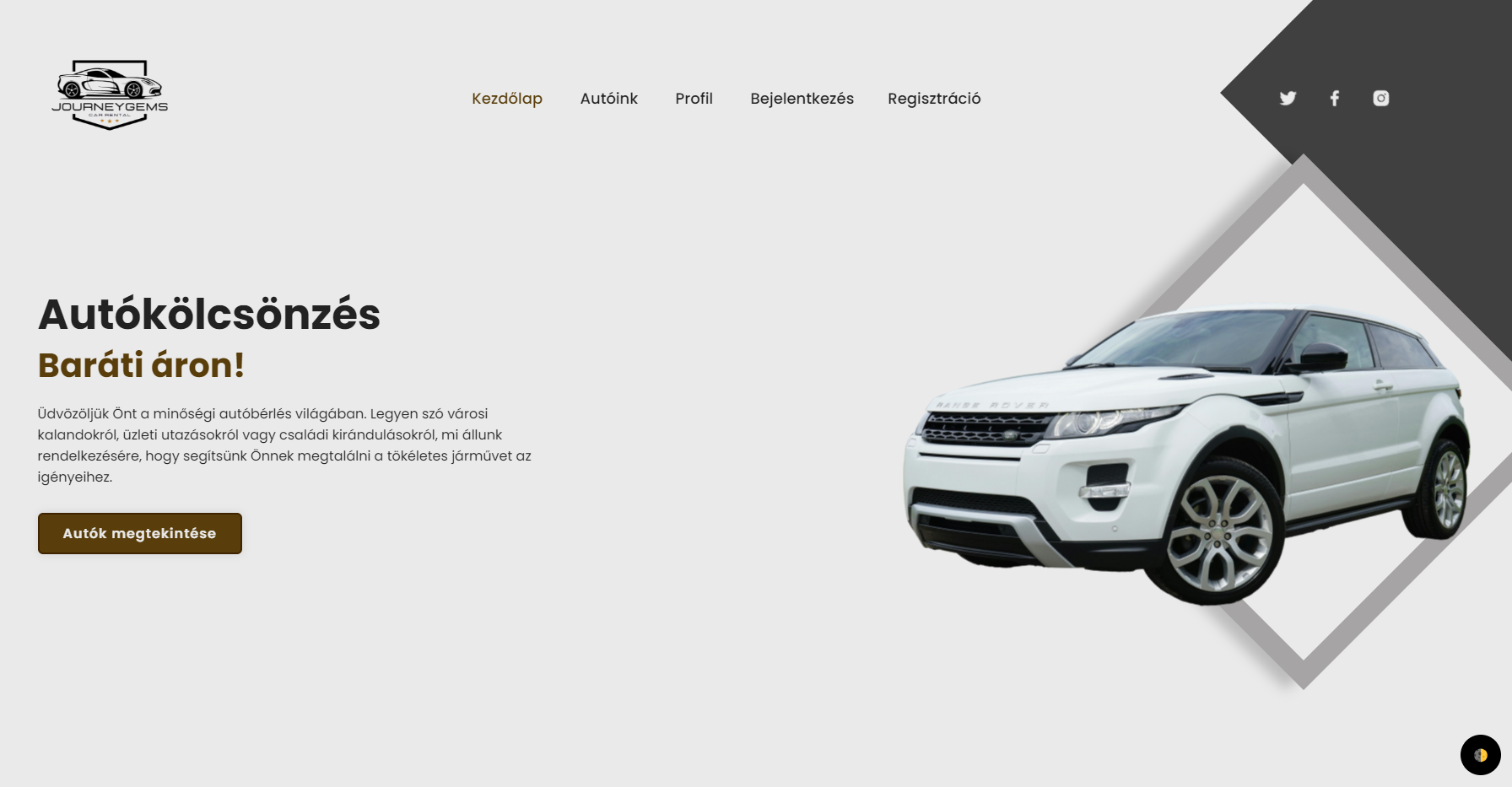
A kezdő oldalunk betöltése során egymást követően két oldalról, illetve felülről késleltetve úsznak be a különböző elemek, animációk alkalmazásának a segítségével, nemcsak egyszerűen megjelenik a teljes weboldal.

A weboldal felső sorában láthatja a tervezett vállalkozásunk logóját, amelyet mi terveztünk meg egy logó tervező alkalmazás segítségével.

A logó mellett egy sorban pedig a navigációs sáv jelenik meg, valamint a közösségi oldalak ikonjai, amelyek jelenleg még csak az adott közösségi oldalra vezetnek, mivel még nem hoztuk létre a vállalkozás önálló oldalait.

A navigációs menünket úgy oldottuk meg, hogy éppen amelyik menüponton vagyunk, azt barna színnel jeleníti meg az oldal, jelezve hogy éppen melyik menüponton állunk.

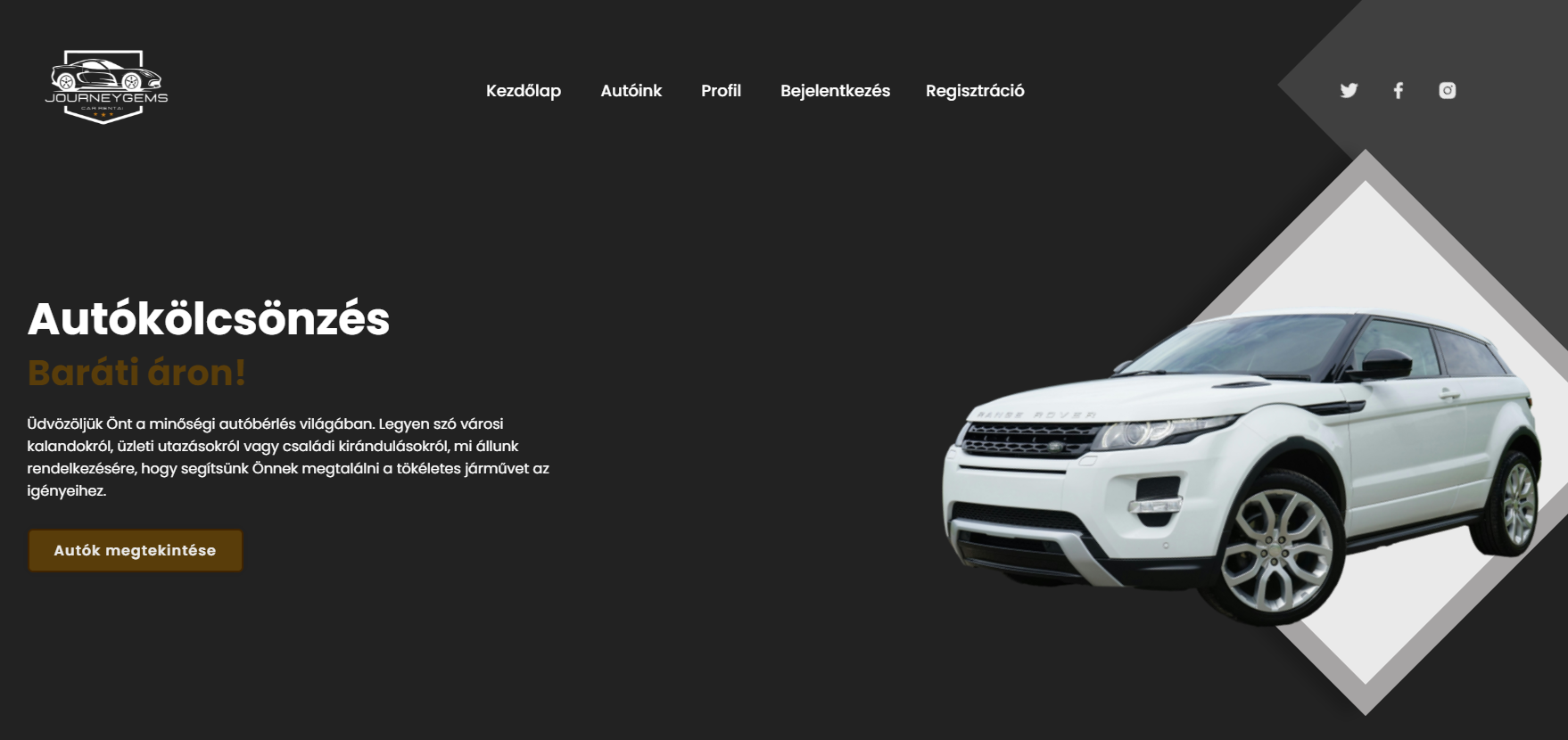
A fő tartalmi részben a baloldalon, egy rövid ajánló látható, alatta, pedig ha autók megtekintése gombra kattint, akkor az elvezeti a bérelhető autók oldalára.



1. ábra Kezdő oldal világos mód

A jobb alsó sarokban egy kis fekete kör alakú ikon látható, amelyre ha a felhasználó rákattint, akkor sötét módra váltja át az oldal megjelenését, de alapértelmezetten a világos megjelenéssel töltődik be a weboldalunk.

Az alábbi képernyő képen ugyanazt az autót látjuk a sötét mód képernyőjén is, de ha a felhasználó frissíti az oldalt, akkor mindig más és más autó jelenik meg, oly módon, hogy 5 előre megadott képből választ ki véletlenszerűen egy-egy új képet.

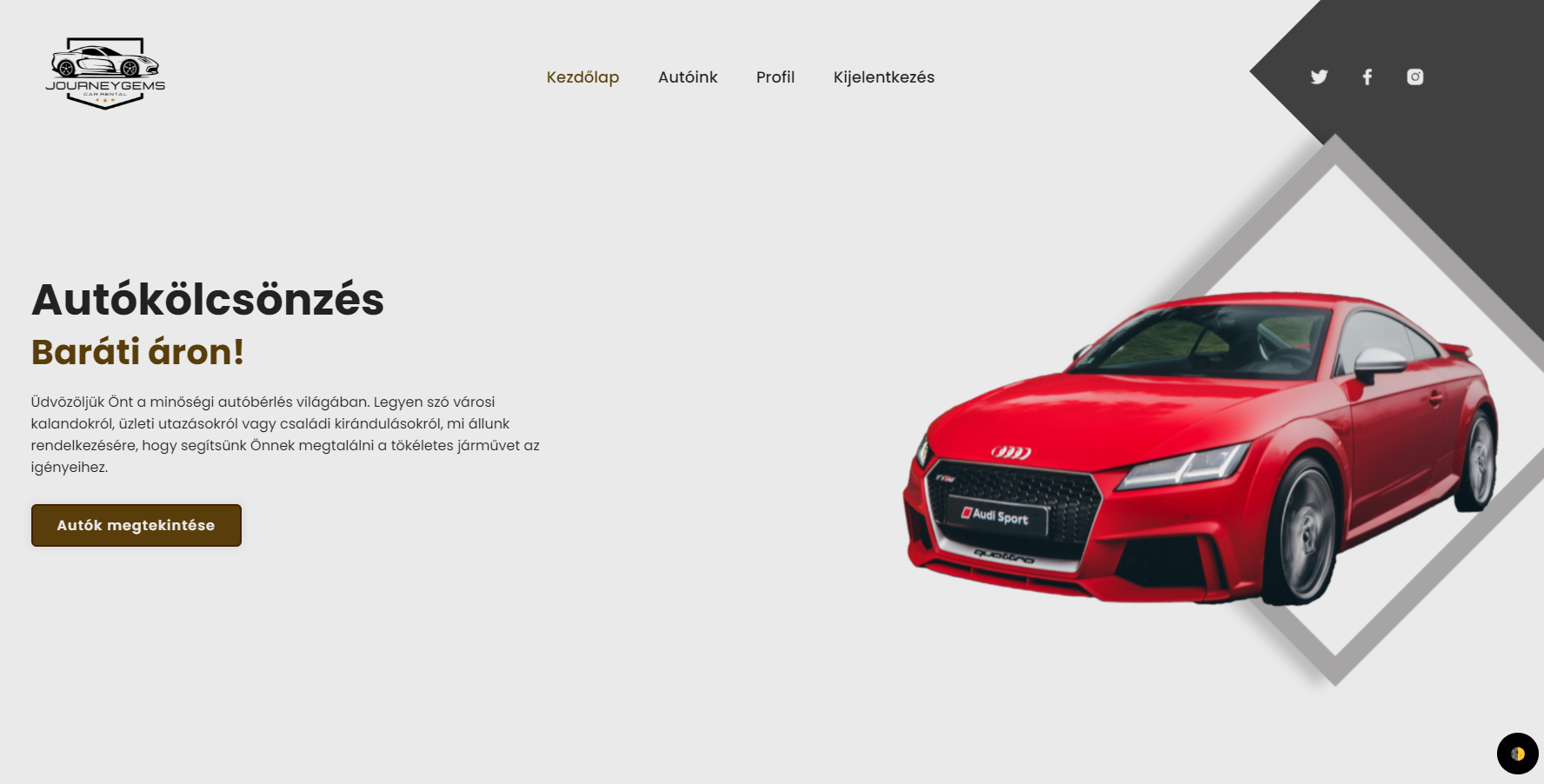


2. ábra Kezdő oldal sötét mód

## A menü

Az oldalakon megjelenő menüt úgy alakítottuk ki, hogyha a felhasználó nincs bejelentkezve, akkor a menüpontok végén a bejelentkezés és regisztráció menüpontok jelennek meg. Mint az előző pontban bemutatott ábrákon (1-1. és 1.2) is látható.

Ha a felhasználó már regisztrált, és bejelentkezett az oldalra, akkor a menü is megváltozik, megjelenik a kijelentkezés menü pont, a bejelentkezés és regisztráció menü pontok helyett. Ezzel is igyekeztünk még inkább felhasználó baráttá tenni az oldalunkat.



3. ábra Bejelentkezés utáni menüpont

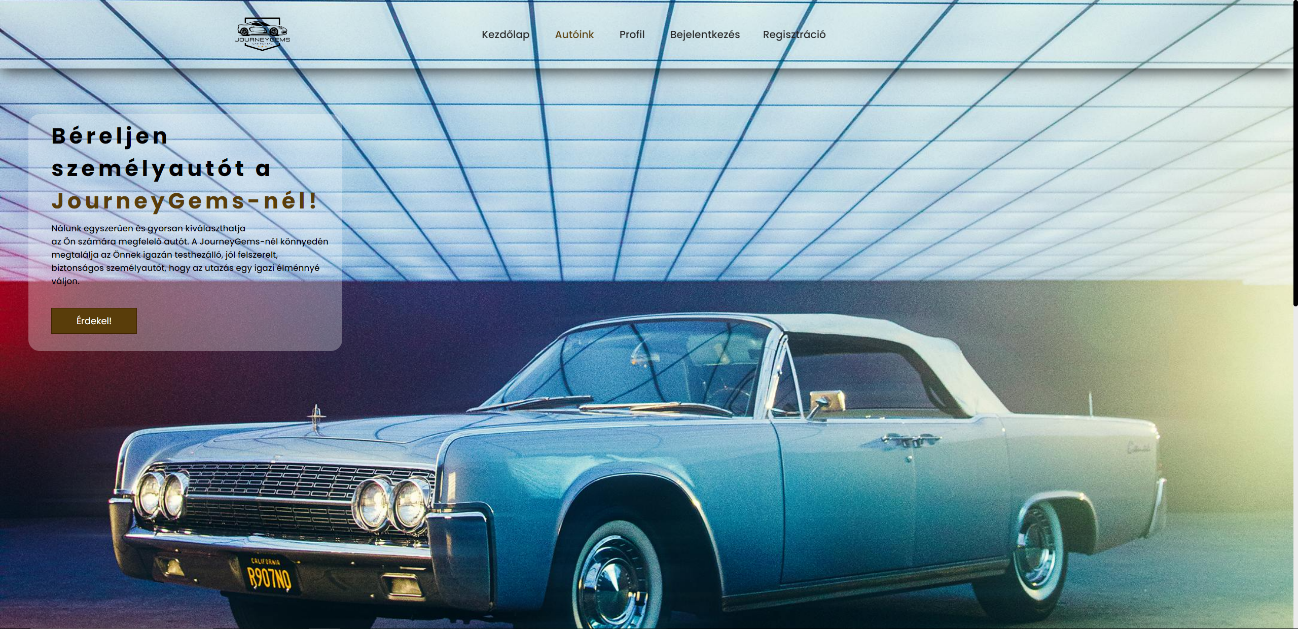
Ha a kijelentkezésre kattint a felhasználó, azonnal kijelentkezteti a rendszer, és a kezdő oldalra navigálja vissza.

## Bérelhető autóink oldala

Ha a kezdő oldal menüjében található Autóink menüpontra, vagy az oldal bal oldalán található Autók megtekintése gombra kattintunk, akkor betöltődik a bérelhető autóink oldala.

Itt is ugyanaz a menü jelenik meg, mint a kezdő oldalon, annyi különbséggel hogy itt egy ún. Paralax technikával oldottuk meg a menü hátterének megjelenését, így próbáltuk érdekesebbé, látványosabbá tenni oldalunkat. A navigációs menü mögött látható az oldalon elhelyezett képünk.

Itt kap helyet egy rövid ajánlónk, illetve egy „Érdekel!” feliratú gomb, amire ha rákattintunk, akkor az oldal alján lévő autóinkhoz görgeti le a felhasználót.



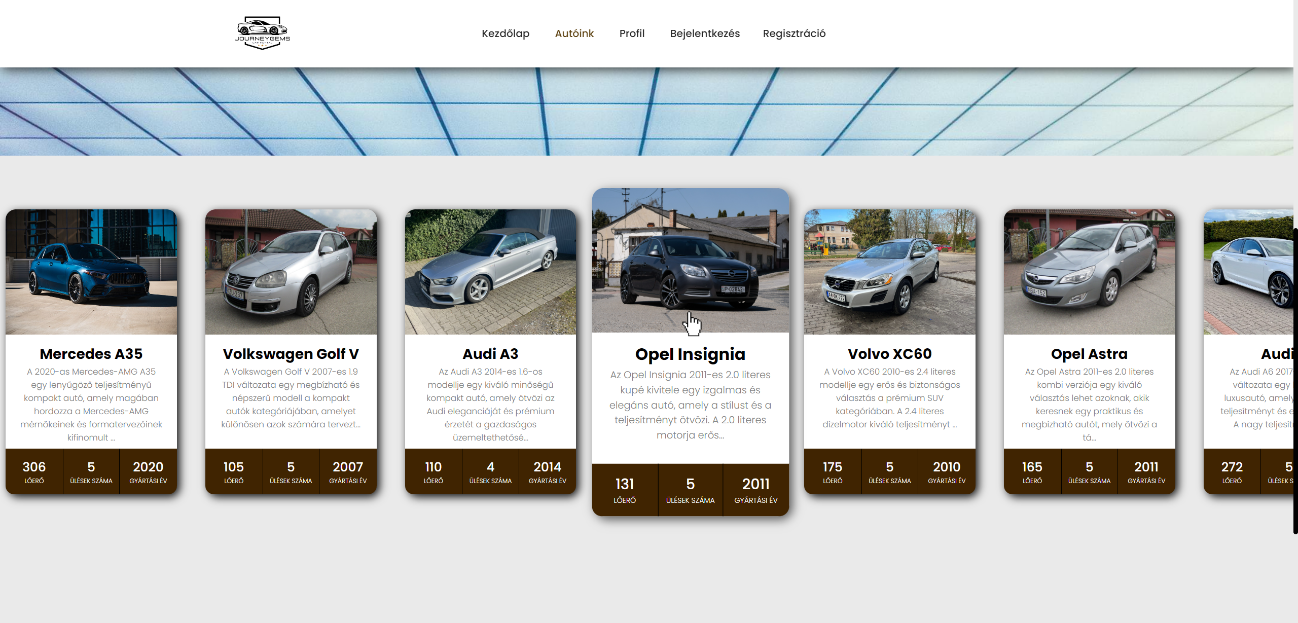
4. ábra Bérelhető autóink oldalának felső része

Ha lejjebb görgetünk, a menü az előre fixált pozíciójában az oldal tetején marad, ugyanakkor a háttérszíne fokozatosan fehér színre vált át, annak függvényében, amennyit lefelé görgettünk az oldalon.

Ha az oldalunk közepére érkezünk, akkor szembe találjuk magunkat a bérelhető autóinkkal. Ha a felhasználó bármelyik autó kártyájára viszi az egerét, akkor az a kártya mérete a képernyőn megnő, ezzel is jobban kiemeltük annak képét illetve leírását, és adatait. Továbbá a felhasználónak még lehetősége van arra, hogy ezeknek a kártyáknak a sávját egérgomb hosszan tartása közben jobbra-balra tudja húzni ezt a sávot, ezzel még több autót megjelenítve.



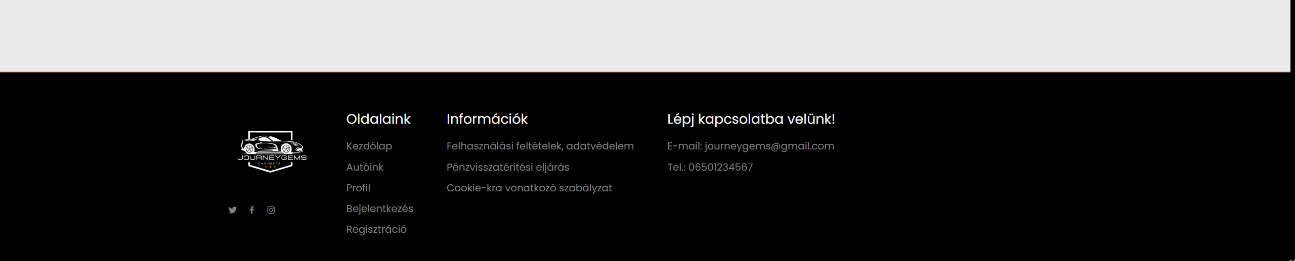
5. ábra Bérelhető autóink



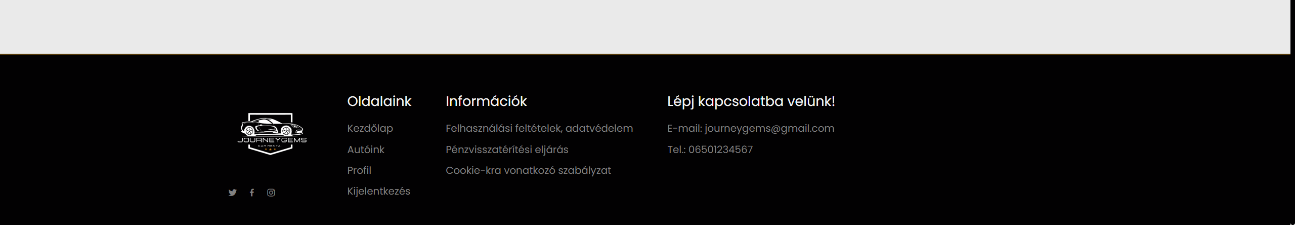
6. ábra Még több bérelhető autó jobb oldalra elcsúsztatva

Ha a felhasználó rákattint valamelyik autóra, akkor az oldal átirányít annak az autónak a foglalós oldalára, ahol még több képet, illetve információt talál arról az adott autóról.

Autóink oldalának legalján még megtalálható egy lábjegyzék (footer), amin az oldalaink menüpontjait (ami ugyanúgy működik, mint a navigációs menünk), egy logót, ami a kezdőlapra visz át, illetve egyéb hasznos információkat helyeztünk el.



7. ábra Lábjegyzékünk nem bejelentkezett felhasználónak

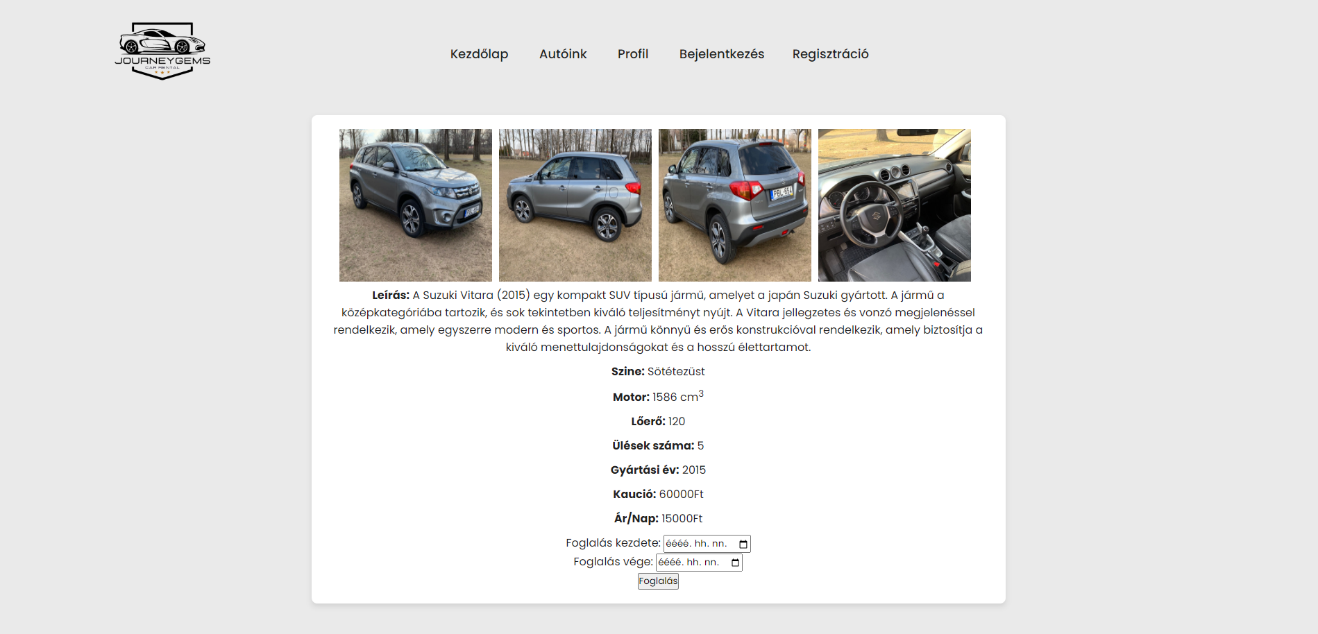


8. ábra Lábjegyzékünk a már bejelentkezett felhasználónak

## Foglalós oldala a kiválasztott autónak

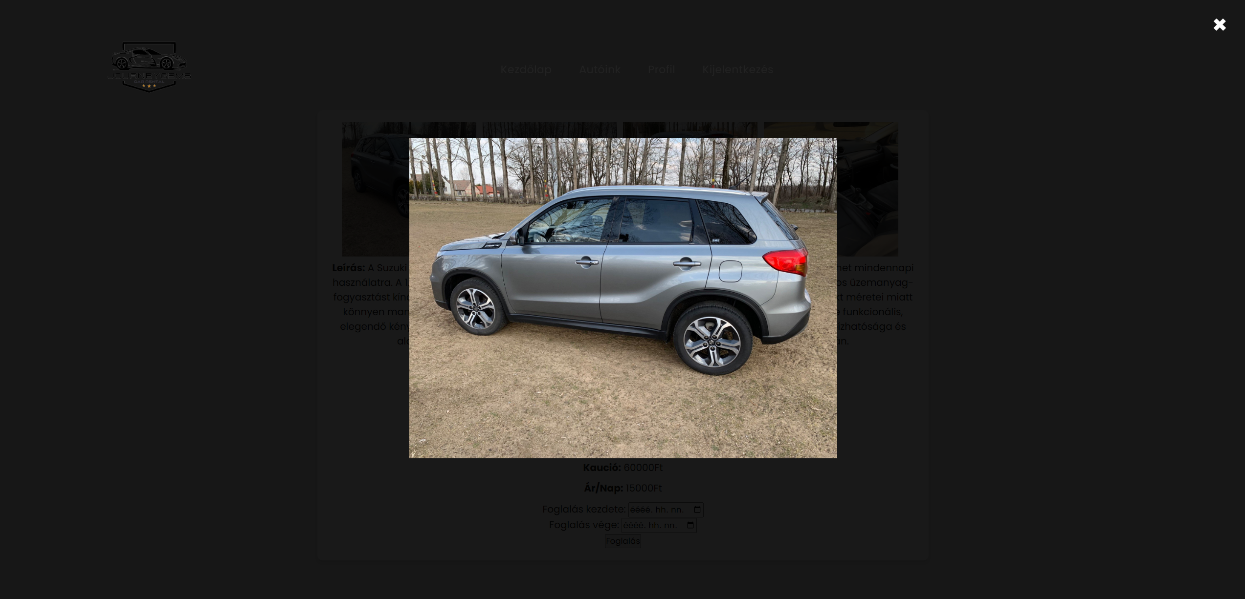
Ha a felhasználó rákattint valamelyik autó kártyájára, akkor annak az autónak egy részletesebb oldala töltődik be, ahol a felhasználó le tudja foglalni az adott autót.

Ezen az oldalon is maradtunk a megszokott stílussal, viszont itt az oldal közepén megjelenik egy fehér doboz, amiben az autóról találunk még további 3 darab képet, illetve az autóról egy hosszabb, részletesebb leírást, illetve adatait. Itt találjuk még az autónak a kaució és a napi árát.



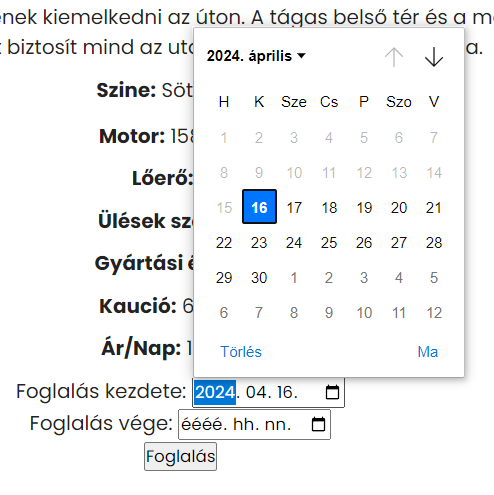
9. ábra Kiválasztott autó oldala

Ha rákattintunk valamelyik képre, akkor az az oldalon nagyban fog megjelenni a jobb átláthatóság érdekében. Ilyenkor a kép középen jelenik meg, körülötte a még látható oldal elsötétül. A képről való kilépést a felhasználó a jobb felső sarokban lévő X-re kattintással tudja megtenni.



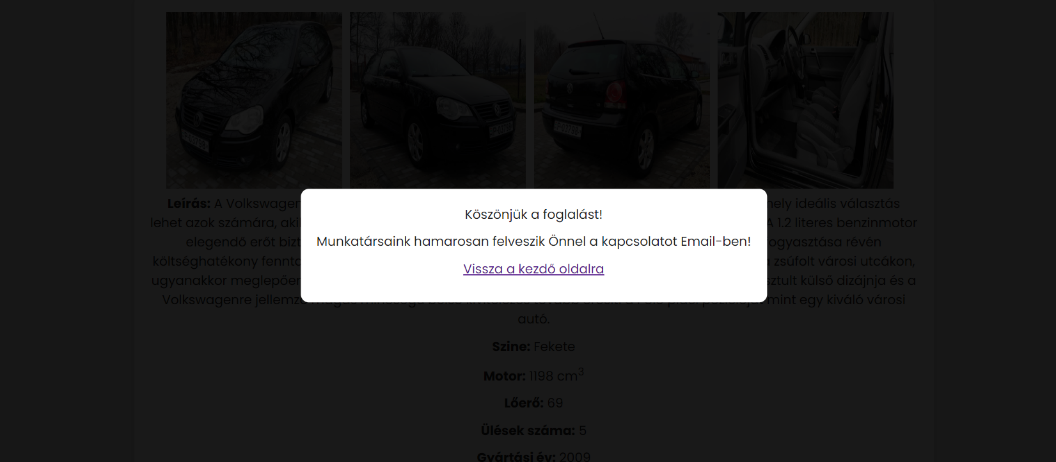
10. ábra Nagyobb kép a foglalós oldalon

Az oldalon megtalálhatunk még két mezőt, illetve egy gombot, amivel a foglalásunkat tudjuk leadni. A két mezőben a foglalásnak a kezdő, és végző időpontját tudjuk megadni. Ha a mezőkben lévő kis naptár ikonra kattintunk, akkor felugrik egy kis naptár ablak, ahol könnyedén kiválaszthatjuk a kívánt napot.



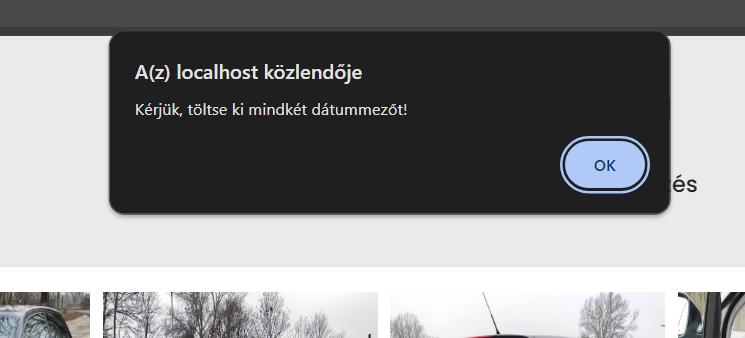
11. ábra Naptár ablak

Ha kiválasztottuk a foglalás kezdetét és végét, akkor a foglalás gombra kattintva egy előugró ablak jelenik, meg ahol tájékoztatjuk a felhasználót a foglalás sikerességéről.



12. ábra Sikeres foglalás

Viszont, ha úgy kattintunk a foglalás gombra, hogy még előtte nem töltöttük ki a mezők valamelyikét, vagy esetleg mind a kettőt, akkor a foglalás nem megy végbe, és ilyenkor az oldal tetején megjelenik egy felugró ablak, amiben tájékoztatjuk a felhasználót, hogy mind a két mezőt ki kell töltenie.

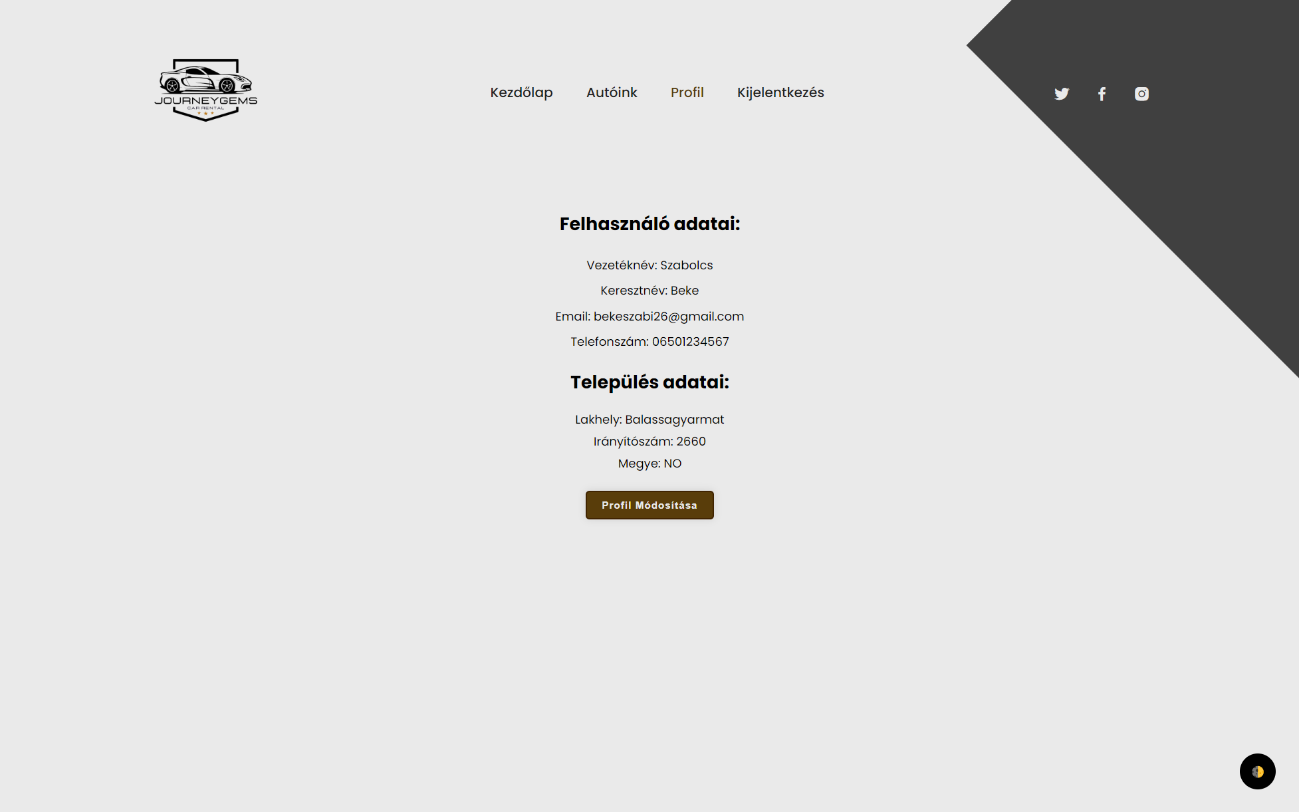


13. ábra Hibás foglalás

## Profil Oldal

Profil oldalunkon maradtunk a kezdő oldal mintájával, ugyanúgy maradt az oldal tetején a navigációs menü, illetve az oldal bal alsó sarkában a sötét mód váltó gomb. Az oldal közepén láthatjuk a felhasználó adatait, alatta pedig a települése adatait, egyértelműen mindet adatbázisból beolvasva AngularJS segítségével.

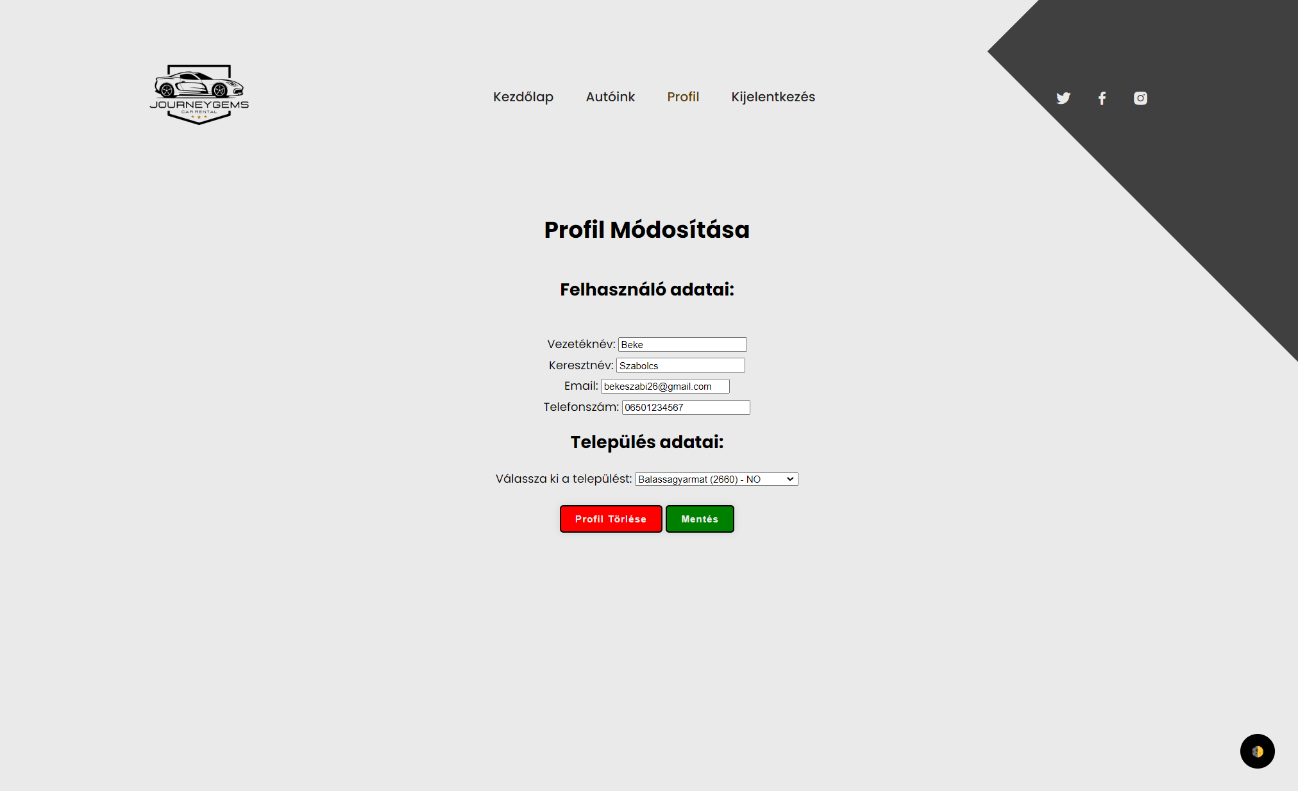
Az oldal viszont csak akkor töltődik be, ha a felhasználó be van jelentkezve, ha nincs bejelentkezve, és úgy kattint a profil oldalra akkor a bejelentkezési oldalra irányítja őt.



14. ábra Profil oldal

Ha a profil módosítása gombra kattint a felhasználó, akkor megváltozik az oldal felépítése, a menü és a sötét mód váltó gomb megmarad, viszont az adatok helyett mezők jönnek, létre ahol a felhasználó beírhatja megváltozott adatait, illetve ezek alatt egy legördülő menüből választhat Magyarország összes településéből abc sorrendben, ha esetleg ezen szeretne változtatni. A település mellett láthatjuk annak irányítószámát és megye rövidítését.

A legördülő menü alatt két gombot találunk, a piros profil törlése gombbal a felhasználó törölni tudja a fiókját, a zöld mentés gombbal pedig a megváltoztatott adatait tudja elmenteni, mentés után pedig az oldal tetején megjelenik egy tájékoztató szöveg, hogyha a mentés sikeres volt, illetve akkor is, ha nem volt sikeres.

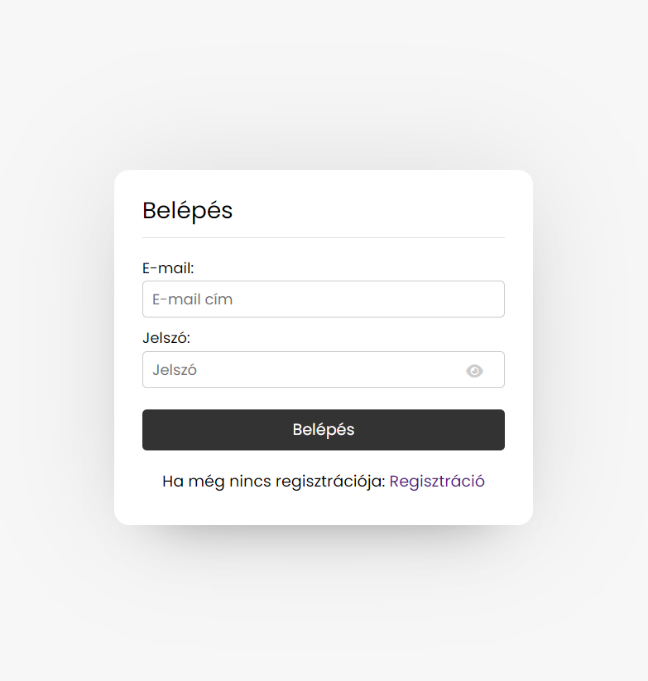


15. ábra Profil módosítása

## Bejelentkezés

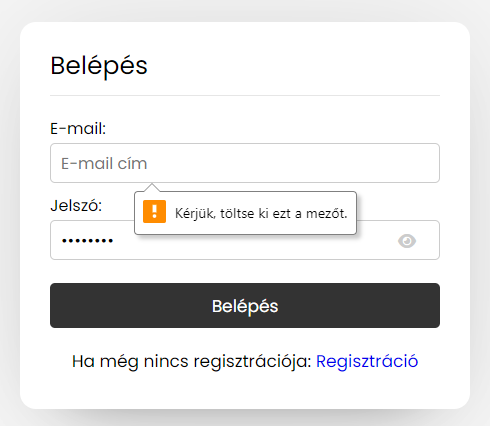
Bejelentkezési oldalunk lényegre törő, nincs túlbonyolítva, szimplán egy ablak jelenik meg előttünk, ahol a felhasználó beírhatja a már meglévő fiókja email címét illetve jelszavát. A jelszó alapból titkosítva jelenik meg ha a felhasználó elkezdi beírni, viszont a kis szem ikonra kattintva ezt láthatóvá tudja tenni. Ha a felhasználó a belépési adatok közül bármelyiket is elírja akkor az oldal nem fogja őt tovább engedni.

Ha esetleg a felhasználónak még nincs profilja akkor innen is könnyedén átléphet a regisztrációs oldalra, a belépés gomb alatti regisztrációs szövegre kattintva.

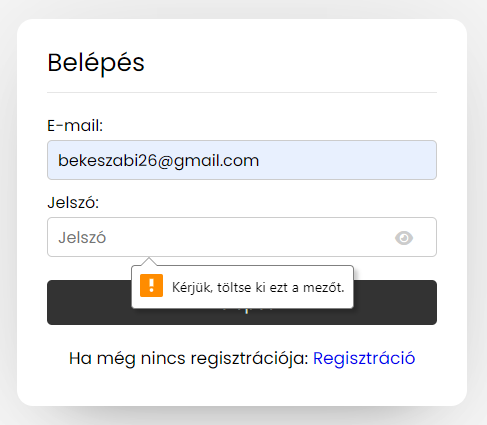


16. ábra Bejelentkezés

Viszont ha a felhasználó véletlenül nem tölti ki valamelyik mezőt, akkor az oldal hibával tér vissza, ami tájékoztatja őt mind a két mező kitöltésére.



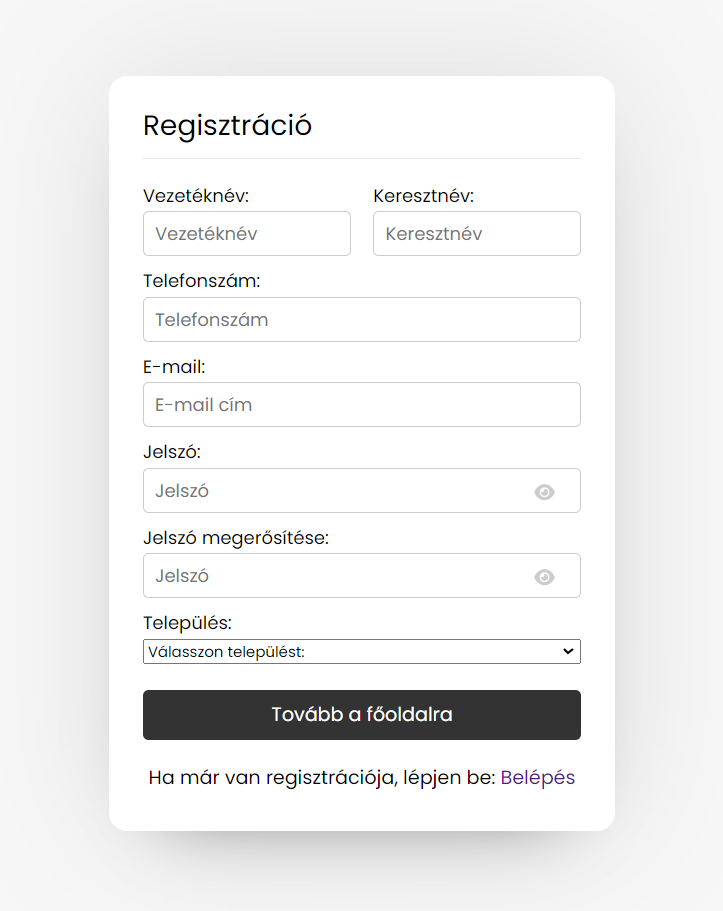
17. ábra Bejelentkezés email nélkül



18. ábra Bejelentkezés jelszó nélkül

## Regisztráció

A regisztrációs oldalunk is szintén könnyedén használható, átlátható. Ha a felhasználó valamelyik mezőt üresen hagyja, vagy az email mezőbe nem ír kukacot, vagy a jelszó és jelszó megerősítése mező nem egyezik akkor az oldal hibát fog kiírni és a regisztráció nem megy végbe. Települését pedig ugyan úgy legördülő menüből választhatja ki.



19. ábra Regisztráció

# Fejlesztői dokumentáció

A projektünkben igyekeztünk minden technológiát felhasználni, amellyel a tanév során megismerkedtünk. Így a munkánk kezdetén az adatok átadására még Javascriptet használtunk, de ahogy később tanultunk az AngularJs-ről, annak az alkalmazását is beépítettük a dolgozatunkba. A backend programozását natív PHP-ben oldottuk meg.

## Felhasznált technológiák

### VSC

A Visual Studio Code (VS Code) egy ingyenesen elérhető, nyílt forráskódú fejlesztői környezet (IDE), melyet a Microsoft fejlesztett ki. Kiemelkedően népszerűvé vált a fejlesztők körében széles körű támogatottsága, kiterjeszthetősége és könnyű használhatósága miatt.

Az VS Code sokféle programozási nyelvet támogat, beleértve a JavaScript-et, Python-t, C#-t és még sok más nyelvet. Kiemelkedő funkciói közé tartozik a beépített Git integráció, amely lehetővé teszi a verziókezelési műveleteket közvetlenül az IDE-ben, valamint a kiterjesztések által nyújtott sokoldalú támogatás, amelyek segítségével testreszabhatóvá tehető a felhasználói élmény.

### HTML

Az HTML (HyperText Markup Language) az internetes tartalmak strukturálására és formázására szolgáló alapvető jelölőnyelv. Az HTML az internetes oldalak alapját képezi, és a böngészők által értelmezett nyelv, amely segít megjeleníteni a weboldalakat a felhasználók számára.

### CSS

A CSS, vagyis a Cascading Style Sheets (Kaskád Stíluslapok), egy stílusleíró nyelv, amelyet weboldalak megjelenítésének szabályozására használunk. A CSS lehetővé teszi a webfejlesztők számára, hogy külön kezeljék a tartalom struktúráját (HTML vagy hasonló jelölőnyelvek által definiált) és a megjelenítés módját. Segítségével beállíthatók a szövegek színei, betűtípusai, elemek elrendezése, margók, keretek és sok más vizuális elem, anélkül, hogy közvetlenül be kellene avatkozni a tartalom kódjába.

### PHP

A PHP (PHP: Hypertext Preprocessor) egy széles körben elterjedt, nyílt forráskódú szkriptnyelv, amelyet elsősorban webes alkalmazások fejlesztésére használnak. A PHP kód a szerveroldalon fut le, ami azt jelenti, hogy a PHP szkripteket a web szerver processzálja, és csak az eredményt küldi el a kliensnek, így a kliens oldalon futó böngésző csak a generált HTML kódot látja.

A PHP-t különösen dinamikus weboldalak és webes alkalmazások létrehozására tervezték. Segítségével adatbázis-műveletek hajthatók végre, munkamenet-kezelés (session management) valósítható meg, cookie-k kezelhetők, és még sok más szerveroldali logika implementálható, ami lehetővé teszi az interaktív és felhasználóbarát weboldalak készítését.

### JavaScript

A JavaScript egy dinamikus, magas szintű, interpretált programozási nyelv, amelyet eredetileg weboldalak interaktív elemeinek létrehozására fejlesztettek ki. Az évek során azonban kiterjesztették a funkcionalitását, így ma már szerveroldali alkalmazások fejlesztésére, mobilalkalmazásokra, játékokra és akár mesterséges intelligencia projektekre is használják.

A JavaScriptet a böngészők natívan támogatják, ami lehetővé teszi, hogy dinamikus tartalmat adjunk a weboldalakhoz anélkül, hogy újra kellene tölteni azokat. Ilyen dinamikus elemek lehetnek például formák adatellenőrzései, animációk, interaktív térképek, élő frissítések és sok más.

### .NET C#

A C# egy típusbiztos, objektumorientált programozási nyelv, amelyet a Microsoft fejlesztett ki, mint a .NET keretrendszer részét. A C# egyike azon nyelveknek, amelyeket a leggyakrabban használnak .NET alapú alkalmazások fejlesztésére, és erőteljes támogatást nyújt a modern programozási paradigmákhoz, mint az objektumorientált programozás (OOP), az eseményvezérelt programozás és az aszinkron programozás.

### XAMPP

A XAMPP célja egy könnyen telepíthető és konfigurálható fejlesztői környezet biztosítása, amely lehetővé teszi a fejlesztőknek, hogy lokálisan fejlesszék és teszteljék webes alkalmazásaikat anélkül, hogy szükség lenne internetkapcsolatra vagy valódi szerverre. A XAMPP használata különösen előnyös, amikor PHP vagy MySQL alapú webalkalmazásokat fejlesztünk, mivel az egész szerverkörnyezetet egyetlen telepítéssel hozhatjuk létre és indíthatjuk el. A XAMPP könnyen kezelhető vezérlőfelülete segítségével a fejlesztők könnyen elindíthatják és leállíthatják a szerverkomponenseket, valamint ellenőrizhetik a naplókat és a konfigurációs beállításokat.

### Microsoft Word

A Microsoft Word egyike a legelterjedtebb szövegszerkesztő programoknak, melyet a Microsoft fejlesztett ki. A szoftver számos funkciót kínál, amelyek lehetővé teszik a felhasználók számára, hogy könnyedén létrehozzanak, szerkeszthessenek és formázhassanak szöveges dokumentumokat.

### Microsoft Excel

Az Excelt általában adatok tárolására, szervezésére és elemzésére használják, legyen az egyszerűbb számlálásoktól és pénzügyi tervezésektől kezdve az összetett adatelemzésekig. Az Excelben táblázatokat hozhatsz létre, amelyek segítenek strukturálni az adatokat és könnyen olvashatóvá teszik azokat. Emellett diagramokat és grafikonokat is készíthetsz az adatok vizuális reprezentálására, ami segíthet a trendek és összefüggések felfedezésében.

### MySQL

A MySQL-t gyakran alkalmazzák olyan webes alkalmazásokban, mint például tartalomkezelő rendszerek, online áruházak, fórumok és sok más, ahol adatbázis-kezelésre van szükség. Az adatok hatékony tárolása, lekérdezése és kezelése mellett a MySQL számos előnyt kínál, többek között a gyors teljesítményt, megbízhatóságot és könnyű kezelhetőséget. A MySQL széles körű támogatást élvez a fejlesztői közösség és dokumentáció révén, ami hozzájárul az elterjedtségéhez és népszerűségéhez a fejlesztők körében.

### GitHub

A GitHub egy webes alapú platform és verziókezelő rendszer, amelyet főként szoftverfejlesztők használnak a kódbázisok kezelésére és közös munkára. A GitHub lehetővé teszi a fejlesztők számára, hogy együtt dolgozzanak a különböző projekteken, nyomon kövessék a változtatásokat, javítsák és bővítsék azokat.

A GitHubot széles körben alkalmazzák a verziókezelésre, amely segít a fejlesztőknek követni és kezelni a kódbázisuk változásait időben. Emellett a GitHub lehetővé teszi a felhasználók számára, hogy hozzájáruljanak mások projektjeihez, problémákat jelentsenek, kódot osszanak meg és tanuljanak más fejlesztőktől. A platform számos eszközt kínál a csapatmunka támogatására, beleértve a probléma- és feladatkezelést, a hozzászólásokat és az áttekintéseket, valamint a kódbázisokhoz kapcsolódó automatizált teszteket és munkafolyamatokat.

### Google Drive

A Google Drive egy felhőalapú tárhely-szolgáltatás és fájlmegosztó platform, amelyet a Google fejlesztett ki és üzemeltet. A felhasználók számára ingyenesen biztosít tárolóhelyet az interneten, amelyen keresztül könnyen lehet elérni és megosztani fájlokat.

A Google Drive lehetővé teszi a felhasználók számára, hogy különböző típusú fájlokat, például dokumentumokat, képeket, videókat és egyéb médiafájlokat tároljanak és szinkronizáljanak különböző eszközeik között, például számítógépek, okostelefonok és tabletek között.

### Discord

A Discord egy ingyenes, platformfüggetlen hang- és szöveges kommunikációs alkalmazás, amelyet elsősorban a játékosok és közösségek használnak. A Discord lehetővé teszi a felhasználók számára, hogy létrehozzanak vagy csatlakozzanak szerverekhez, ahol cseveghetnek, hanghívásokat indíthatnak, videókat oszthatnak meg, valamint tartalmakat és élményeket oszthatnak meg egymással.

### dbdiagram.io

A dbdiagram.io egy online eszköz, amelyet adatbázis-diagramok készítésére és tervezésére használnak. Ez a platform lehetővé teszi a felhasználók számára, hogy vizuálisan tervezzenek és modellezzenek adatbázisokat, könnyen megérthető diagramokat készítsenek az adatbázis tábláiról, kapcsolatairól és attribútumairól.

A dbdiagram.io hasznos eszköz fejlesztők, adatbázis-adminisztrátorok és tervezők számára, akiknek szükségük van egy átlátható módon megjelenített adatbázis-struktúrára. A platform segítségével a felhasználók könnyedén létrehozhatnak, módosíthatnak és megoszthatnak adatbázis-diagramokat, ami megkönnyíti az együttműködést és a kommunikációt a csapatok között az adatbázis-tervezés folyamatában.

### AngularJS

Az AngularJS egy JavaScript keretrendszer, amelyet elsősorban a front-end webfejlesztéshez használnak. Széles körű népszerűségét a könnyű használhatóságával, a dinamikus webalkalmazások készítésének lehetőségével és a nagyszerű dokumentációjával érte el. Az AngularJS különlegessége, hogy az adatok és a felhasználói felület közötti kapcsolatot tisztán és hatékonyan kezeli, ezáltal lehetővé téve a dinamikus és reszponzív webalkalmazások fejlesztését.

## Adatbázis készítésének lépései

### Az adatbázis célja

Adatbázisunk fő célja, hogy hatékonyan kezelje és tárolja az összes olyan információt, amely a weboldalunk működéséhez szükséges. Az adatbázisban tárolt adatok közé tartoznak az autóink adatai, a bérlők adatai illetve a foglalások adatai. Az adatbázis lehetővé teszi az összefüggő adatok összekapcsolását így könnyen nyomon követhető, hogy melyik ügyfél melyik autót bérelte és mikor.

### Tervezés megkezdése

Számos autóbérlő vállalkozásának weboldalát tanulmányoztuk, hogy milyen koncepció alapján építsük fel az adatbázisunkat, milyen táblákat hozzunk létre, milyen oszlopokkal, az oszlopok alá milyen típusú rekordok, illetve idegen kulcsok kerüljenek be.

### Tervezési lépések

Az adatbázis szerkezetét a DB Diagram weboldalán ([dbdiagram.io](https://dbdiagram.io/d)) készítettük el, ami jelentősen megkönnyítette a tervezés lépéseit. Egyszerűen létrehozható a saját új diagrammunk egyetlen kattintással, ahol kapunk egy kezdő sablon mintát, amelyet tovább tudtunk alakítani. Az oldal bal oldal sávján azonnal megjelennek a létrehozott táblák, alatta pedig a hozzá adott idegen kulcsok. A jobb oldalon pedig az adatbázis diagrammjai és a táblák kapcsolatai.

### Egyedtípusok/egyedek meghatározása

* **Renter (Bérlő)** – A Renter egyed a bérlővel kapcsolatos információkat, elérhetőségét tartalmazza.
* **Settlement (Lakcím)** – A Settlement egyed a települést, illetve annak irányítószámát tárolja.
* **Car (Autó)** – A Car egyed az autóink információit tartalmazza.
* **Brand (Márka)** – A Brand egyed az adott autónak a márkáját tárolja. Pl.: Suzuki.
* **CarModel (Modell)** – A CarModel egyed az adott autóról tároja a modell tipusát. Pl.: Vitara.
* **BodyType (Kivitel típus)** – A BodyType egyed az adott autóról tároja a kivitele tipusát.
* **FuelType (Üzemanyag típus)** – Az FuelType az adott autónak tárolja az üzemanyag tipusát.
* **DriveType (Hajtás típus)** – A DriveType tábla az adott autóról tárolja a hajtásának tipusát.
* **TransmissionType (Váltó típus)** – A TransmissionType az adott autóról tárolja a váltójának tipusát.

### Kapcsolatok meghatározása

* A **Car** és **Renter** tábla között **N:M** kapcsolat van mert, egy autót több bérlő is kibérelhet, és egy bérlő több autót is kibérelhet.
* A **Renter** és **Settlement** tábla közötti kapcsolat: **N:1**, mivel egy bérlő egy településhez, és egy településhez több bérlő kapcsolható.
* A **Car** és **BodyType** tábla közötti kapcsolat: **N:1**, mivel egy autó egy kivitelhez tartozhat, de egy kivitelhez több autó is kapcsolódhat.
* A **Car** és **FuelType** tábla közötti kapcsolat: **N:1**, mivel egy autó egy üzemanyagtípushoz tartozhat, viszont egy üzemanyagtípushoz több autó is tartozhat.
* A **Car** és **Brand** tábla közötti kapcsolat: **N:1**, mivel egy autó egy márkához tartozhat, viszont egy márkához több autó is kapcsolódhat.
* A **CarModel** és **Brand** tábla közötti kapcsolat: **N:1**, mivel egy modellhez egy márka tartozhat, de egy márkához tartozhat több modell is.
* A **Car** és **DriveType** tábla közötti kapcsolat: **N:1**, mivel egy autóhoz egy hajtástípus tartozik viszont, egy hajtástípushoz több autó is tartozhat.
* A **Car** és **TransmissionType** tábla közötti kapcsolat: **N:1**, mivel egy autóhoz egy sebességváltótípus kapcsolódhat, de egy sebességváltótípus több autóhoz is tartozhat.

### N:M kapcsolatok felbontása

* **Reservation (Foglalás)** – A Reservation egyed az adott autónak a foglalásáról és a foglalóról tartalmazza az információkat.
* A **Reservation** kapcsolótábla felbontása:
* A **Renter** és **Reservation** tábla közötti kapcsolat: **1:N**, mivel egy ügyfél több foglalást is készíthet, de egy foglalás csak egy ügyfélhez tartozik.
* A **Reservation** és **Car** tábla közötti kapcsolat: **N:1**, mivel egy foglalás egy autóhoz tartozik, de egy autóhoz több foglalás is tartozhat.

### Táblák

**Renter (Bérlő) tábla**

|  |  |  |  |
| --- | --- | --- | --- |
| **mező neve** | **mező típusa** | **mező leírása** | **megjegyzés** |
| RenterID | integer(11) | Bérlő azonosítója | elsődleges kulcs |
| SettlementID | integer(11) | Bérlő lakcím azonosítója | idegen kulcs, Settlement táblához kapcsolódik |
| Firstname | varchar(30) | Bérlő keresztneve |  |
| Surename | varchar(30) | Bérlő vezetékneve |  |
| TelephoneNumber | varchar(12) | Bérlő telefonszáma |  |
| Email | varchar(50) | Bérlő e-mail címe |  |
| Password | varchar(255) | Bérlő jelszava titkosítva |  |

**Reservation (Foglalás) tábla**

|  |  |  |  |
| --- | --- | --- | --- |
| **mező neve** | **mező típusa** | **mező leírása** | **megjegyzés** |
| ReservationID | integer(11) | Foglalás azonosítója | elsődleges kulcs |
| RenterID | integer(11) | Bérlő azonosítója | idegen kulcs, Renter táblához kapcsolódik |
| CarID | integer(11) | Autó azonosítója | idegen kulcs, Car táblához kapcsolódik |
| StartingDay | date | Bérlés kezdő időpontja |  |
| EndingDay | date | Bérlés végző időpontja |  |
| TimeStamp | timestamp | Bérlés foglalásának időpontja |  |

**BodyType (Autó típus)**

|  |  |  |  |
| --- | --- | --- | --- |
| **mező neve** | **mező típusa** | **mező leírása** | **megjegyzés** |
| BodyTypeID | integer(11) | Kivitel azonosító | elsődleges kulcs |
| BodyTypeName | varchar(15) | Kivitel neve |  |

**FuelType (Üzemanyag típus)**

|  |  |  |  |
| --- | --- | --- | --- |
| **mező neve** | **mező típusa** | **mező leírása** | **megjegyzés** |
| FuelID | integer(11) | Üzemanyag azonosító | elsődleges kulcs |
| FuelName | varchar(10) | Üzemanyag típus |  |

**Brand (Márka )**

|  |  |  |  |
| --- | --- | --- | --- |
| **mező neve** | **mező típusa** | **mező leírása** | **megjegyzés** |
| BrandID | integer(11) | Márka azonosító | elsődleges kulcs |
| BrandName | varchar(12) | Márkanév |  |

**CarModel (Modell)**

|  |  |  |  |
| --- | --- | --- | --- |
| **mező neve** | **mező típusa** | **mező leírása** | **megjegyzés** |
| ModelID | integer(11) | Modell azonosító | elsődleges kulcs |
| BrandID | integer(11) | Márka azonosító | idegen kulcs, Brand táblához kapcsolódik |
| CarModelName | varchar(10) | Modell neve |  |

**DriveType (Meghajtás típus)**

|  |  |  |  |
| --- | --- | --- | --- |
| **mező neve** | **mező típusa** | **mező leírása** | **megjegyzés** |
| DriveID | integer(11) | Meghajtás azonosító | elsődleges kulcs |
| DriveName | varchar(3) | Meghajtás neve |  |

**TransmissionType (Váltó típus)**

|  |  |  |  |
| --- | --- | --- | --- |
| **mező neve** | **mező típusa** | **mező leírása** | **megjegyzés** |
| TransmissionID | integer(11) | Váltó azonosító | elsődleges kulcs |
| TransmissionType | varchar(10) | Váltó típus |  |

**Car (Autó)**

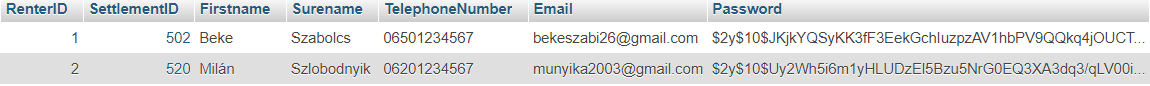
|  |  |  |  |
| --- | --- | --- | --- |
| **mező neve** | **mező típusa** | **mező leírása** | **megjegyzés** |
| CarID | integer(11) | Autó azonosítója | elsődleges kulcs |
| ModelID | integer(11) | Modell azonosító | idegen kulcs, CarModel táblához kapcsolódik |
| BodyTypeID | integer(11) | Kivitel azonosító | idegen kulcs, BodyType táblához kapcsolódik |
| FuelID | integer(11) | Üzemanyag azonosító | idegen kulcs, FuelType táblához kapcsolódik |
| DriveID | integer(11) | Meghajtás azonosító | idegen kulcs, DriveType táblához kapcsolódik |
| TransmissionID | integer(11) | Váltó azonosító | idegen kulcs, TransmissionType táblához kapcsolódik |
| Description | varchar(1500) | Leírás |  |
| Picture | varchar(255) | Fénykép | Fájlnév pl.: kép.jpg |
| Color | varchar(10) | Szín |  |
| Engine | varchar(20) | Motor |  |
| Performance | varchar(6) | Teljesítmény |  |
| Seats | int(11) | Ülések |  |
| ProductionYear | int(11) | Gyártási év |  |
| AirConditioning | tinyint(4) | Klíma | 0: Nincs klíma az autóban  1: Van klíma az autóban |
| Deposit | int(11) | Kaució |  |
| PricePerDay | int(11) | Ár/nap |  |

**Settlement (Település)**

|  |  |  |  |
| --- | --- | --- | --- |
| **mező neve** | **mező típusa** | **mező leírása** | **megjegyzés** |
| SettlementID | integer(11) | Cím azonosító | elsődleges kulcs |
| SettlementName | varchar(20) | Település neve |  |
| ZipCode | varchar(4) | Irányítószám |  |
| County | varchar(2) | Megye monogrammja |  |

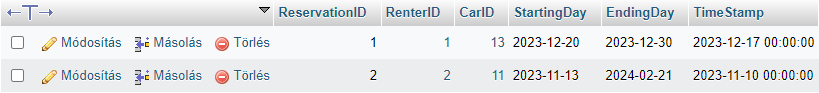
### Adatbázis ábrák

**Renter (Bérlő)**



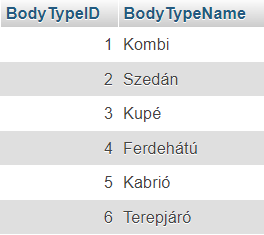
20. ábra Renter tábla adatokkal

**Reservation (Foglalás)**



21. ábra Reservation tábla adatokkal

**BodyType (Kivitel)**



22. ábra BodyType tábla adatokkal

**FuelType (Üzemanyag tipus)**



23. ábra FuelType tábla adatokkal

**Brand (Márka)**



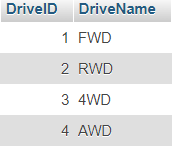
24. ábra Brand tábla adatokkal

**CarModel (Modell)**



25. ábra CarModel tábla adatokkal

**DriveType (Kivitel)**

****

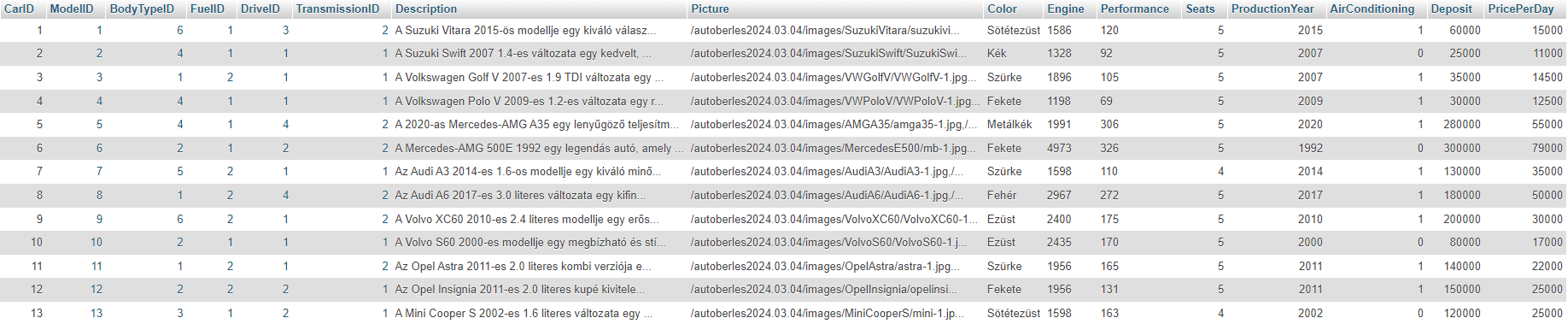
26. ábra DriveType tábla adatokkal

**TransmissionType (Váltótipus)**

****

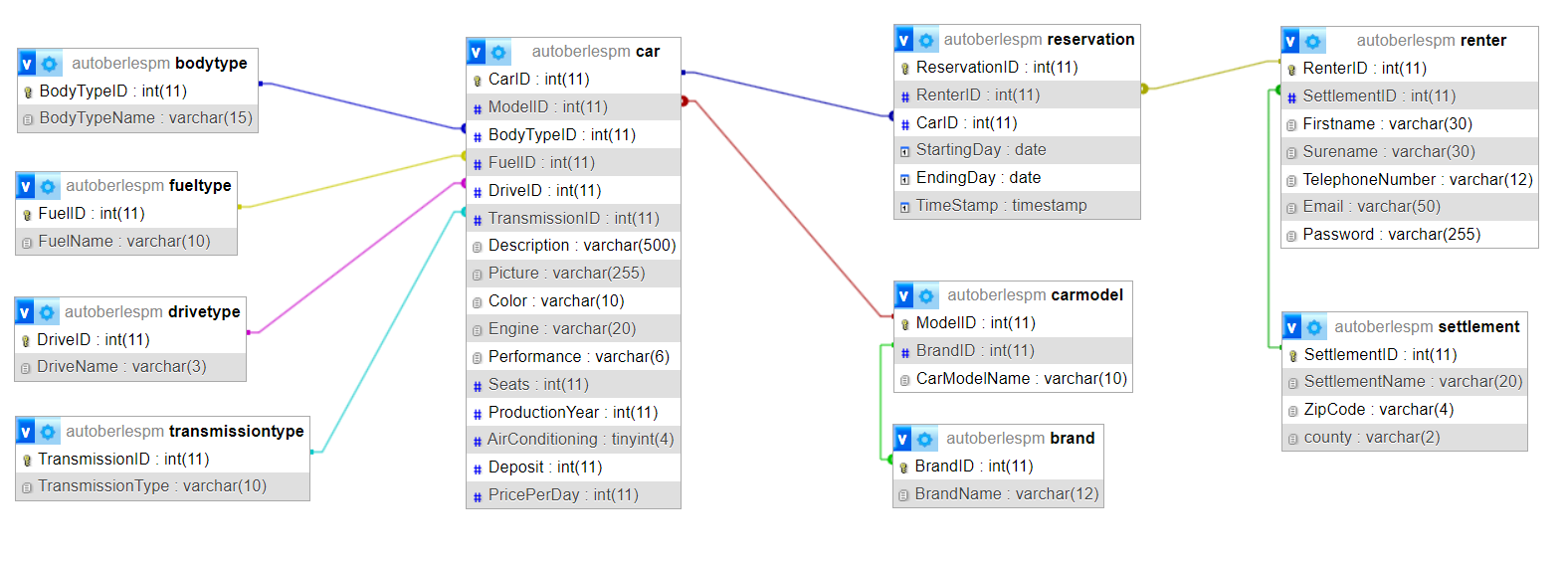
27. ábra TransmissionType tábla adatokkal

**Car (Autó)**

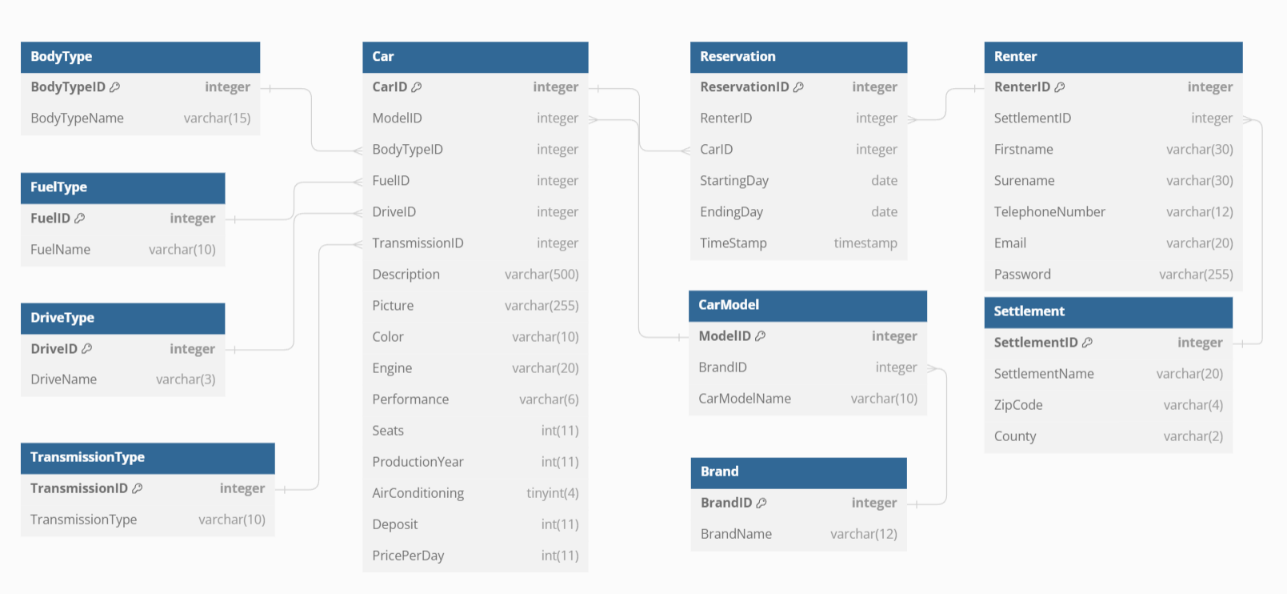
****

28. ábra Car tábla adatokkal

### Adatbázis diagram



29. ábra phpMyAdmin tervező felülete



30. ábra dbdiagram.io tervező felülete

## Frontend programozás

### Kezdőoldal

A kezdőoldalunk az index.php fájlban került eltárolásra. A navigációt úgy programoztuk, hogy a menü pontok egymás után jelenjenek meg a weboldalon 1 másodperces késéssel. Ehhez minden menüpontnak meghatároztuk az értékét, hogy mennyivel később töltődjenek be.

## Backend programozás

A projektünkben a Frontendet és Backend programozást igyekeztünk ketté választani, a frontend oldalak feldolgozását JavaScript-el, illetve AngularJS-el továbbítottuk a backend mappában elhelyezett fájlok felé. A backend fájljainkkal JSON formátumú végpontokat hoztunk létre, amelyeket AngularJS segítségével jelenítettük meg a frontend oldalon.

### Adatbázis kapcsolat létrehozása

Az adatbázissal való kapcsolat a „config.php” fájlba került kiszervezésre, azért, hogy újra fel lehessen használni, és ne kelljen minden alkalommal újra hozzá adni az adott oldalakhoz, amikor adatbázis kapcsolatra van szüksége a programnak.

A kapcsolat létrehozásához a *„mysqli\_connect()”* függvényt használtuk és azonnal el is tároltuk a *„$conn”* néven deklarált változóba.

Ezt követően egy egyszerű elágazás segítségével ellenőriztük, hogy létre jön-e a kapcsolat. Miután a weboldalon megjelent a „kapcsolat létrejött” felirat, az ellenőrzést komment jelek közé zártuk.

Végül a mysqli\_query függvény segítségével beállítottuk az adatbázis karakterkódolását UTF-8-ra. Ez azért fontos, mert nemcsak olyan szöveget szeretnénk megjeleníteni, amelyek az angol ábécének felelnek meg, hanem ékezetes betűket is. Az UTF-8 karakterkódolás, pedig lehetővé teszi a különböző karakterkészletek kezelését, így az ékezetes betűk megfelelő megjelenítését is.

### Regisztrációs adatok feldolgozása

A signup.php fájl futtatásával a böngészőben megjelenik a regisztrációs űrlap, amelyet, ha a felhasználó kitölt, és elküld, továbbításra kerül feldolgozásra. A tervezés során készített signup.html fájlt először is átneveztem signup.php-re, és az include\_once parancssal csatoltam a fejlécet. Hozzá adtam a munkamenethez a session\_start(); függvény segítségével, amelynek az a funkciója, hogy az aktuális munkamenetet elindítja, vagy a meglévőt folytatja.

Lényegében a felhasználói adatok tárolására szolgál a szerver és a böngésző között. A session\_start() függvényt minden olyan oldalhoz csatolnom kellett, ahol a munkamenetet használni szükséges, mivel segít a PHP-nek azonosítani a felhasználót és a hozzá tartozó aktuális munkamenetet.

Az if(isset($\_SESSION['user\_id'])): elágazással megvizsgálom, hogy létezik e már belépett felhasználó, azaz a $\_SESSION tömbben létezik-e egy olyan kulcs, amely 'user\_id' névvel rendelkezik. Ha ez a kulcs létezik, az azt jelenti, hogy a felhasználó már belépett, mivel a 'user\_id' munkamenetváltozó a bejelentkezés után kerül beállításra.

Ha az adott feltétel igaz, azaz a 'user\_id' munkamenetváltozó létezik, a kód az header("location:users.php"); sorral tovább navigálja a felhasználót a "users.php" nevű oldalra.

Az űrlap adatainak feldolgozására létre hoztam a „regisztracio.js” fájlt, amely a DOM (Dokumentum Objektum Modell) elemeket manipulálja. A fájlban olvasható kódom célja a „signup.php” oldalam DOM-jában található elemek elérésére és kezelésére szolgál, melynek segítségével továbbítani tudom az űrlap adatait a „backend/ signup.php” oldalra feldolgozás céljából.

A regisztracio.js fájlomban első lépésként deklaráltam a signupForm nevű konstans változót, melyben eltároltam php oldalon összeállított űrlapot, a küldés gombot és a hibaüzenetek megjelenítésére szolgáló konténert. Az elemek kiválasztáshoz a querySelector() metódust alkalmaztam.

Ezt követően a signupForm változóhoz hozzá kötöttem egy onsubmit eseménykezelőt, amennyiben a felhasználó elküldi az űrlap adatait, a preventDefault() metódussal megakadályozom a böngésző alapértelmezett működését, vagyis az oldal újratöltését és az űrlap adatainak továbbítását a szerver felé. Erre azért van szükség, mivel el akartam kerülni az űrlap elküldésekor az oldal újbóli betöltését, mivel az adatok továbbítását JavaScript segítségével oldottam meg.

A küldéssel egy időben a küldés gombhoz hozzá kötöttem egy onclick esemény kezelőt, amely a kattintásra elindítja az AJAX kérést. Létre hoztam a XMLHttpRequest új objektum példányát, amely lehetővé teszi az aszinkron HTTP kérések küldését a szerver felé.

Konfiguráltam az újonnan létrehozott objektumot, ahol a POST metódus végzi a kérés küldését a "backend/signup.php" URL-re. A megadott harmadik paraméter, a „true” az aszinkron működést jelzi. Az „onload” eseménykezelő akkor fut le, amikor a kérés befejeződik.

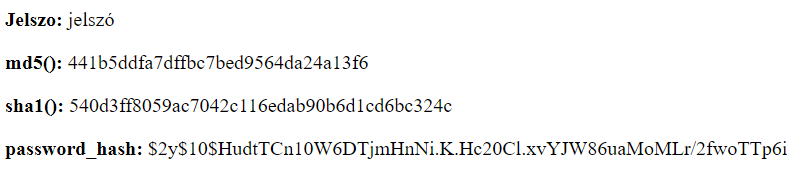
A következő kódblokk az ellenőrzéseket tartalmazza, mint befejeződött-e a kérés, 200-as-e a státusza, amely azt jelenti sikeres volt a kérés. A válasz a data változóba kerül mentésre, amelynek, ha az értéke a sikeres küldéssel egyenlő, akkor a böngésző átirányítja a felhasználót az "index.php" oldalra. Ellenkező esetben a hibaüzenet megjelenítésre kerül az errorText változóban és a weboldalon.

Végezetül az űrlap adatait el kell küldenünk az Ajax segítségével a PHP szervernek. Ehhez először létrehoztam a FormData objektumot, amely az űrlap adatait tartalmazza. Majd ezt követően elküldtem az űrlap adatait a szervernek a XMLHttpRequest segítségével.

A backend/signup.php fájlban történik az űrlap adatainak a feldolgozása. A létrehozott változókba eltároltam az adatokat. A SQL beinjektálás kivédésre, a mysqli\_real\_escape\_string() függvényt használtam. Minden olyan weboldalon, ahol lehetőséget adunk a felhasználóknak adatok felvitelére, fontos az SQL injektálás elleni védelem, ugyanis a rosszindulatú felhasználó SQL paranccsal átveheti az oldal felett az irányítást, vagy hozzáférhet az adatbázis bizalmas adataihoz, illetve törölheti azokat.

A jelszó esetében a biztonságos tárolás érdekében hash függvény használatát alkalmaztam. A hash függvények (kiejtése: hes, magyarul hasítófüggvények) olyan eljárások, amelyekkel bármilyen hosszúságú adatot adott hosszúságra képezhetünk le. Az így kapott véges adat neve hash/hasító érték. Ezek az algoritmusok az 1980-as évek legvégén az elektronikus aláírás megjelenésével váltak szükségessé.

A titkosításra többféle módszer is született az évek folyamán, az első között az md5() egy 32 karakterből álló, az sha1() egy 40 karakterből álló, a password\_hash() pedig egy 60 karakterből álló titkosított jelszót állít elő, amelyet nagyon nehéz vissza fejteni. Ezek eredményét a php/passwordtest.php oldalamon mutatom be, miként tárolja el a „jelszó” szót a különböző titkosítási metódusokkal.



31. ábra php/passwordtest.php oldal

Dolgozatomban a password\_hash() függvény használatával végeztem el a jelszavak titkosítását, amely a PHP beépített funkciója, amely két paramétert vár. Az első paraméterben az átadott jelszót fogja hashelni azaz titkosítani, a második paraméterben pedig a hashelési algoritmus kiválasztása szerepel. A PASSWORD\_DEFAULT konstans azt jelenti, hogy a PHP az aktuális legjobb gyakorlatoknak megfelelő algoritmust választja ki automatikusan a titkosított jelszó generálásához.

Az interneten számos oldalt találhatunk, amelyek a kiválasztott titkosítási módszerrel legenerálják a jelszót, és ugyanígy találhatunk visszafejtési opciókra is megoldásokat.

Az űrlap elemeinek eltárolása után, megvizsgálom minden adat megadásra került-e. Az első if-állítással megvizsgálom, hogy az összes szükséges mező ($Surename, $Firstname, $TelephoneNumber, $Email, $Password) kitöltésre került e, azaz nem üres. Ha bármelyik változó érték nélkül maradt, akkor a "Minden mezőt ki kell töltenie!" hibaüzenet jelenik meg a felhasználónak.

Ha az összes mező kitöltve van, akkor következik az e-mail cím formátumának ellenőrzése a filter\_var függvény segítségével, ugyanakkor a mysqli\_query() segítségével lekérdezést hajtok végre, nem került e már be a megadott e-mail cím az adatbázisba. Ha már létezik az adatbázisban, akkor a felhasználó ismét hiba üzenetet kap.

Ha az e-mail cím érvényes formátumú és nem létezik az adatbázisban, akkor a program megkezdi profil feltöltésének feldolgozását.

### Beléptetés

A már regisztrált felhasználók a regisztrációs oldalról is átléphetnek egy kattintással a belépésre, vagy a főoldalról is elérhetik a belépés oldalát. A beléptetéshez ugyanúgy, mint a regisztrációhoz 3 fájlt hoztam létre. Az első, amely a felhasználó előtt megjelenik a login.php oldala, ahol meg kell adni a felhasználó nevet és a jelszó a dolgozónak az input mezőkben. Ehhez az oldalhoz is csatoltam a pass-show-hide.js fájlt, ugyanúgy, mint a regisztrációs oldal esetében, amellyel felfedhetővé válik a beírt jelszó, valamint a login.js fájlt, melynek kódja lényegében megegyezik a signup.js fájl tartalmával, ugyanis itt is AJAX kérés segítségével továbbítom az adatokat a backend/login.php oldalnak feldolgozásra.

Ez esetben is először eltárolom a frontend oldalról érkező adatokat, majd megvizsgálom, hogy az e-mail cím létezik-e, az adatbázisban, ha nem létezik a felhasználó hibaüzenetet kap. Az $sql változóban egy SQL lekérdezést definiáltam, amely a renter táblában keres olyan rekordot, ahol az email mező tartalma megegyezik a $writtenemail változó értékével. A LIKE operátor abban segít, hogy részleges illeszkedést is elfogadásra kerüljön az e-mail cím esetében.

A mysqli\_query($conn, $sql) függvényhívással indítottam el a lekérdezés futtatását az adatbázisban. Az eredményt a $result változóban tároltam el.

Ezt követően a mysqli\_num\_rows($result) függvény megszámolja, hány sor található az adatbázisban, a $result eredményhalmazon belül. Ha nincs megfelelő sor az adatbázisban, akkor az "Hibás e-mail címet adott meg!" üzenet jelenik meg a felhasználónak, majd a return; utasítás megszakítja az további programfutást.

Az if elágazással megvizsgáltam, hogy a $email és $writtenpass változók nem üresek-e. Ha valamelyik üres, akkor a "Hibás bejelentkezési adatok." üzenet jelenik meg, majd a kód végrehajtása leállításra kerül.

Ha az adatok kitöltésre kerültek, akkor a program ellenőrzi, hogy a megadott jelszó megegyezik e, a tárolt hash értékkel. Ehhez a password\_verify függvényre volt szükségem, amely egy beépített PHP függvény, amelyet a paramétert vár, az első érték a felhasználó által megadott jelszó, vagyis a bejelentkezési kísérlet során beírt jelszó. A második érték az adatbázisban eltárolt titkosított 60 karakterből álló jelszó. A password\_verify() függvény, ezeket az értékeket összehasonlítja, és visszatér egy logikai értékkel. Ha a két jelszó egyezik, a true értékkel tér vissza, jelezve, hogy a megadott jelszó helyes. Ha a két jelszó nem egyezik, akkor a false értékkel tér vissza, jelezve, hogy a megadott jelszó helytelen. Ebben az esetben a "Hibás bejelentkezési adatok." üzenet jelenik meg.

### Foglalás

Ha a felhasználó kiválasztott egy autót, a foglalás időpontjait, és a foglalás gombra kattint akkor az adatbázisba beszúrásra kerülnek a reservation táblába a foglalás részletei. A felhasználó ID-jét a SESSION-ból azaz a munkamenetből tároltam el egy változóba. A carID-t a GET kéréssel az URL-ből kapom meg, ugyanis amikor a felhasználó rákattint egy autóra és megnyilík az autó részleteinek oldala akkor az URL-be bekerül a kiválasztott autó ID-je.

<http://localhost/autoberles2024.03.04/autoberles2024main/carDetails.php?carId=4>

A kezdő és a záró dátumot a POST metódus segítségével kapom meg, ha létezik POST, a következő módon:

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

//print\_r("fut"); ellenőrizem hogy lefut a függvény gombkattintásra

$startingDay = $\_POST['startingDay'];

$endingDay = $\_POST['endingDay'];

//print\_r($startingDay); ellenőriztem hogy kezembe kerültek e az adatok

//print\_r($endingDay);

$sql2 = mysqli\_query($conn, "INSERT INTO reservation (RenterID, CarID, StartingDay, EndingDay)

VALUES ('{$renterId}', '{$carId}', '{$startingDay}', '{$endingDay}')");}

Ahogyan a kódrészletemből is látszik, első lépésben ellenőriztem hogyha a foglalás gombra kattintok, akkor végrehatja e a kérésemet hogy írja ki a kérésemet a böngészőbe. Ezt követően deklaráltam a $startindDay, $endingDay változókat amelyekbe eltároltam a két kiválasztott dátumot. Mind kettőt ellenőriztem a print\_r() segítségével hogy sikerült e eltárolnom az adatokat. Amikor meggyőződtem arról hogy mind a 4 adat a kezembe került amelyeket szükséges a reservation táblába beszúrni, végrehajtottam az INSERT INTO parancsal.

### Profil oldal

A mappa szerkezetben létrehoztunk egy profilF.php fájlt, amely megjeleníti a belépett felhasználó adatait. Ehhez a backend mappában a profil.php fájlban történik meg a felhasználó adatainak kiolvasása az adatbázisból, amelyből ezt követően egy végpontot „json” formátumú kimenetet generálunk. A backend mappában lévő fájlokhoz a session\_start() metódussal munkamenetet adtunk hozzá, majd ezt követően megvizsgáljuk hogy van e belépett felhasználó:

if (!isset($\_SESSION['RenterID'])) {

    header("Location: login.php");

    exit();}

tehát hogyha nem létezik a belépett felhasználó ID-je akkor a program automatikusan a Login oldalra irányítja a felhasználót hogy lépjen be. Ezt követően létrehoztuk a $RenterID = $\_SESSION['RenterID']; amelyben eltároltuk a munkamenetbe szereplő felhasználó ID-jét.

Ez alapján az adatbázisból a renter táblából kiolvassuk a felhasználó adatait, mint például lakhely, irányítószám, megyekód, foglalási ID (ha van), foglalt autó ID-je, a foglalás első illetve záró napja.

$sql = "SELECT renter.\*, settlement.SettlementName, settlement.ZipCode, settlement.county, reservation.ReservationID, reservation.CarID, reservation.StartingDay, reservation.EndingDay

        FROM renter

        LEFT JOIN settlement ON renter.SettlementID = settlement.SettlementID

        LEFT JOIN reservation ON renter.RenterID = reservation.RenterID

        WHERE renter.RenterID = $RenterID";

Ehez az adatbázis lekérdezéshez a Left Join parancsot hazsnáltuk mivel 3 táblából kell beolvasnunk az adatokat. A renter, settlement és reservation táblákból.

A LEFT JOIN egy olyan SQL utasítás, amely két táblát összekapcsol, és visszaadja az összes rekordot az első (bal oldali) táblából, valamint a második (jobb oldali) táblából azokat a rekordokat, amelyek megfelelnek a megadott feltételnek. Ha nincs megfelelő rekord a jobb oldali táblában, akkor a LEFT JOIN NULL értékeket ad vissza a jobb oldali tábla oszlopaiban.

$result = mysqli\_query($conn, $sql);

Ebben a kódrészletben a $conn a MySQL adatbáziskapcsolatunkat reprezentálja.

A mysqli\_query() függvény végrehajtja az SQL lekérdezést az adatbázison, és visszaad egy eredményhalmazt, ha a lekérdezés SELECT, SHOW, DESCRIBE vagy EXPLAIN utasítás. Ha a lekérdezés sikeresen végrehajtott egy INSERT, UPDATE, DELETE, vagy REPLACE utasítást, akkor TRUE értéket ad vissza. Ha hiba történt, akkor FALSE értéket ad vissza.

Az eredményhalmazt a $result változóban tároljuk, amit később felhasználhatunk az adatok kiolvasására.

Létrehoztunk egy új üres tömböt: $data = array();

Egy ciklussal végig iteráltunk az adatbázis táblánk sorain, melyhez a mysqli\_fetch\_assoc() függvényt használtuk, amellyel betudjuk olvasni az adatbázis-lekérdezések eredményhalmazának sorait. Ez a függvény egy asszociatív tömböt ad vissza, amelyben a kulcsok az oszlopnevek, és az értékek az adott sorban lévő adatok.

while ($row = mysqli\_fetch\_assoc($result)) {

    $data['renterData'] = array(

        'FirstName' => $row['Firstname'],

        'Surname' => $row['Surename'],

        'Email' => $row['Email'],

        'TelephoneNumber' => $row['TelephoneNumber']

    );

A json fájl előállításához echo json\_encode($data); függvbényt használtuk mely paraméterként megkapta a beolvasott adatokat. A kód a $data változó tartalmát JSON formátumra alakítja, és kiírja a kimenetre.

A json\_encode() függvényt arra használják, hogy PHP értékeket (pl. tömböket vagy objektumokat) JSON formátumú karakterlánccá alakítsa át.

Ezzel lényegében egy API-t állítottunk elő autentikációval, amelyet a link birtokában le tudunk kérdezni ha regisztrált felhasználóként próbáljuk elérni.

Az adatok feldolgozásához már az AngularJS technológiát választottuk, mivel a tanórák folyamán a projekt írás közben ezzel is megismerkedtünk, és mindenképpen alkalmazni akartuk az újjonan tanultakat, mivel sokkal kevesebb kódolással juthatunk hozzá az adatokhoz mint a hagyományos Javascript használatával.

Ehhez a Frontend oldali profilF.php kódját is módosítani kellet. A HTML elemekhez hozzá kellett adnunk a szükséges Angular direktívákat. Elsőként az ng-app direktívát és a ng-controller direktívát. Az elnevezések függvényében készítettük el a profil.js fájlunkban az adatok lekérését az API-ból.

Első lépésben létrehoztunk egy új AngularJS modult renterApp néven: angular.module('renterApp', []);. A második paraméter, a [], a függőségek listája, amelyeket a modul használ. Az app.controller('RenterController', function($scope, $http) {...}); sorban létrehozunk egy új kontrollert RenterController néven. A kontroller a modell és a nézet közötti “közvetítő”. Itt a $scope és $http beépített AngularJS szolgáltatások, amelyeket a kontroller használ.

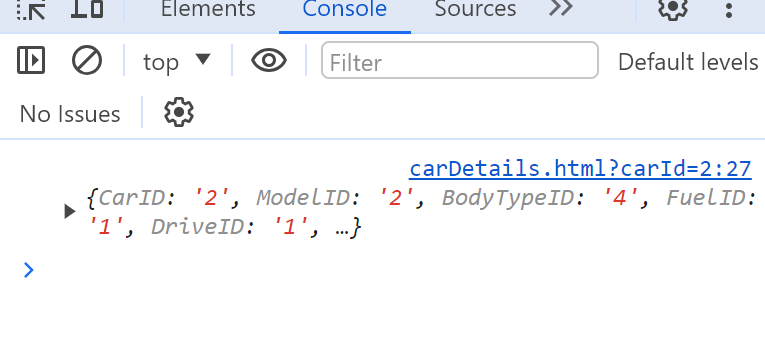
A $http.get('http://localhost/autoberles2024.03.04/autoberles2024-main/backend/profil.php') sorban egy HTTP GET kérést indítunk a megadott URL-re. Ez a kérés aszinkron, tehát a válasz megérkezése után fut le a then(function(response) {...}); blokkban lévő kód.

A response objektum tartalmazza a szerver válaszát. A response.data.renterData, response.data.settlementData és response.data.reservationData sorokban a válasz adatait mentjük el a $scope változóban, amit később felhasználhatunk az adatok megjelenítésére a nézetben.

Végül a console.log($scope.renterData, $scope.settlementData, $scope.reservationData) sorban kiírjuk a konzolra a kapott adatokat.

## Tesztelés

A különböző fájlok programozása során folyamatosan teszteltük a programunk működését. Minden egyes változó deklarálása esetén ellenőriztük, hogy sikerült e az elérni kívánt adatokat eltárolni. A javascript kódok esetében a console.log() beépített függvény segítségével ellenőriztük a böngészőbe kiíratva az adatokat.



32. ábra Javascript kód tesztelése

A PHP kódok fejlesztése során gyakran szükség van arra, hogy megvizsgáljuk a változók vagy adatstruktúrák aktuális állapotát, hogy megbizonyosodjunk arról, hogy a programunk helyesen működik. Erre a célra általában olyan beépített függvényeket használunk, mint például a var\_dump(), print\_r() vagy die(). Ezek a függvények segítenek abban, hogy a kódból adott pillanatban kiírassuk a változók vagy más adatstruktúrák tartalmát.

A var\_dump() függvény kiírja a változók értékét, típusát és méretét is, ami rendkívül hasznos volt a változók részletes elemzéséhez, hogy milyen típusú adatokat kapunk vissza.

while ($row = mysqli\_fetch\_assoc($result)) {

// Minden autó adatait hozzáadja a tömbhöz

$cars[] = $row;

var\_dump($cars); }

A print\_r() függvény pedig a változó vagy tömb struktúráját jeleníti meg olvasható formában, ami különösen hasznos volt, amikor bonyolultabb adatstruktúrákat vizsgáltunk.

A die() függvény lehetővé teszi a kód végrehajtásának azonnali leállítását, és az aktuális üzenet kiírását. Ezt akkor használtuk, amikor az előző két függvény használata nem vezetett eredményre, hogy megállítsuk a kód további végrehajtását valamilyen feltétel teljesülése esetén, a hibakeresés során.

# Összefoglalás

A munkánk során a legnagyobb felmerült akadály a dark mód beállítása volt. Ugyanis találtunk egy nagyon jó, egyszerűnek tűnő megoldást arra, hogy a sötét módot hozzá tudjuk adni az oldalunkhoz, ehhez egy már előre megírt JavaScript könyvtárat a DarkMode.js kellett csatolni, és egy rövidke kódrészlettel kiegészíteni. Egy YouTube videót is találtunk, és mindent az utasítások szerint követtünk, de, a DarkMode mégse működött. Ekkor kipróbáltuk egy üres HTML fájlon, mert arra gondoltunk, hogy valamelyik kódrészletünk felülírja kód működését. Ez a felvetésünk be is igazolódott a teszt során, mivel az üres oldalon kiválóan működött. Ekkor elkezdtük megvizsgálni, hogy az index oldalon melyik az a kód részlet, amely megakadályozza a Dramode.js alapértelmezett működését. Egyesével töröltük az elemeket, mire rájöttünk, hogy a probléma a stílus lapon keresendő. Megvizsgáltuk a DarkMode.js tartalmát, amelyben megtaláltuk, hogy a css tulajdonságok és értékek hol kerültek definiálásra, így ezeket a stílusokat felülírtuk egy külön style3.css fájlban. Végül már csak a fájlok csatolásának sorrendjére kellett figyelnünk, hogy a DarkMode.js után adjuk meg a style3.css való hivatkozást. Így már működésre bírtuk a sötét módot kattintásra, viszont a logó fotójának cseréjét is le kellett programoznunk. A DarkMode gombra való kattintásnál meghívásra került a toggleDarkMode() nevű függvény, amelyben meghívtuk az imgWhite() függvényt, amelyben egy elágazás segítségével létrehoztunk egy kapcsolót, ahol a feltételben megvizsgáljuk a logo source attribútumát, ha a fekete hátterű kép van megadva akkor a fehér hátterű kép jelenjen meg, ha pedig fehér a háttér akkor a fekete színű logó jelenjen meg.

A másik problémánk akkor adódott, amikor úgy döntöttünk, hogy több képet szeretnénk egy autóhoz megjeleníteni. Az adatbázisba 4 kép linkjét adtam meg, amelyeket vesszővel választottam el. Viszont ebben a formában a fő oldalon nem jelent meg a 4 kép közül egy sem. Így az angularJs használatával kezdtem keresni olyan szeletelési megoldást, amely kompatibilis az Angularral. Találtam is, amit tökéletesen be tudtam implementálni a kódomba.

ng-repeat="picture in car.Picture.split(',')"

Ez a PHP és AngularJS kód együttműködve egy olyan funkciót valósít meg, ahol a "car" változó egy objektum, melynek "Picture" nevű mezője egy karakterlánc, amelyben vesszővel elválasztva szerepelnek az autó képeinek elérési útjai. Az "ng-repeat" direktíva segítségével az AngularJS a "car.Picture" karakterláncot vesszők mentén feldarabolja, majd minden darabot a "picture" változóba helyezi, így lehetővé téve egy képgaléria létrehozását az autó képeivel.

# Továbbfejlesztési lehetőségek

* Regisztrációnál vissza igazoló e-mail
* Elfelejtett jelszó visszaállítása
* Kereső optimalizálás a weboldalra
* Precízebb foglalás ellenőrzés
* Foglalás megerősítés emailben
* Admin felület létrehozása, ahol fel lehet tölteni új autókat, valamint lehet módosítani, és törölni is, nemcsak az adatbázison keresztül
* A saját profil felületén megtekinthető foglalások

# Források:

* + 1. WEBiskola: Mi az a PHP? A PHP fogalma és bemutatása Link:

https://webiskola.hu/php-ismeretek/mi-az-a-php-fogalma-bemutatasa/

* + 1. Wikipedia: CSS Link: https://hu.wikipedia.org/wiki/CSS
    2. Wikipedia: GitHub Link: <https://hu.wikipedia.org/wiki/GitHub>
    3. Wikipedia: HTML Link: <https://hu.wikipedia.org/wiki/HTML>
    4. Wikipedia: MySQL Link: <https://hu.wikipedia.org/wiki/MySQL>
    5. Wikipedia: PHP Link: <https://hu.wikipedia.org/wiki/PHP>
    6. Wikipedia: Visual Studio Code Link:

https://hu.wikipedia.org/wiki/Visual\_Studio\_Code

* + 1. Wikipedia: XAMPP Link: <https://hu.wikipedia.org/wiki/XAMPP>
    2. W3School: How TO - Slideshow Link:

<https://www.w3schools.com/howto/howto_js_slideshow.asp>

* + 1. How To Add Dark-mode To Your Website Link: https://www.youtube.com/watch?v=lJmw3zoEt3g
    2. Pexels Stockfotók Link: https://www.pexels.com/hu-hu/
    3. Unsplash Stockfotók Link: https://unsplash.com/
    4. A Complete Guide to CSS Grid Link: <https://csstricks.com/snippets/css/complete-guide-grid/>
    5. A Complete Guide to Flexbox Link: https://css-tricks.com/snippets/css/a-guide-to-flexbox/
    6. CSS Grid Container Link: <https://www.w3schools.com/css/css_grid_container.asp>
    7. Ajax function Link: <https://api.jquery.com/jquery.post/>
    8. CSS-TRICKS - Media Queries for Standard Devices Link: <https://css-tricks.com/snippets/css/media-queries-for-standard-devices/>
    9. Questions tagged [animation] Link: https://stackoverflow.com/questions/tagged/animation

# Ábrajegyzék

[1. ábra Kezdő oldal világos mód 4](#_Toc164237528)

[2. ábra Kezdő oldal sötét mód 5](#_Toc164237529)

[3. ábra Bejelentkezés utáni menüpont 5](#_Toc164237530)

[4. ábra Bérelhető autóink oldalának felső része 6](#_Toc164237531)

[5. ábra Bérelhető autóink 7](#_Toc164237532)

[6. ábra Még több bérelhető autó jobb oldalra elcsúsztatva 7](#_Toc164237533)

[7. ábra Lábjegyzékünk nem bejelentkezett felhasználónak 7](#_Toc164237534)

[8. ábra Lábjegyzékünk a már bejelentkezett felhasználónak 8](#_Toc164237535)

[9. ábra Kiválasztott autó oldala 8](#_Toc164237536)

[10. ábra Nagyobb kép a foglalós oldalon 9](#_Toc164237537)

[11. ábra Naptár ablak 9](#_Toc164237538)

[12. ábra Sikeres foglalás 10](#_Toc164237539)

[13. ábra Hibás foglalás 10](#_Toc164237540)

[14. ábra Profil oldal 11](#_Toc164237541)

[15. ábra Profil módosítása 12](#_Toc164237542)

[16. ábra Bejelentkezés 12](#_Toc164237543)

[17. ábra Bejelentkezés email nélkül 13](#_Toc164237544)

[18. ábra Bejelentkezés jelszó nélkül 13](#_Toc164237545)

[19. ábra Regisztráció 13](#_Toc164237546)

[20. ábra Renter tábla adatokkal 24](#_Toc164237547)

[21. ábra Reservation tábla adatokkal 24](#_Toc164237548)

[22. ábra BodyType tábla adatokkal 24](#_Toc164237549)

[23. ábra FuelType tábla adatokkal 24](#_Toc164237550)

[24. ábra Brand tábla adatokkal 25](#_Toc164237551)

[25. ábra CarModel tábla adatokkal 25](#_Toc164237552)

[26. ábra DriveType tábla adatokkal 25](#_Toc164237553)

[27. ábra TransmissionType tábla adatokkal 25](#_Toc164237554)

[28. ábra Car tábla adatokkal 26](#_Toc164237555)

[29. ábra phpMyAdmin tervező felülete 26](#_Toc164237556)

[30. ábra dbdiagram.io tervező felülete 26](#_Toc164237557)

[31. ábra php/passwordtest.php oldal 30](#_Toc164237558)

[32. ábra Javascript kód tesztelése 34](#_Toc164237559)