

**Szakképesítés neve:** Szoftverfejlesztő és tesztelő

**Szakmai azonosító száma:** 5 0613 12 03

# VIZSGAREMEK

Autóbérlés megvalósítása weboldalon

**Készítette:** Székely József

# Tartalomjegyzék

Bevezetés.....	3
1. Felhasználói dokumentáció.....	4
1. 1 Felhasználói követelmények.....	4
1.1.1 Általános elvárások.....	4
1.1.2 Specifikus elvárások.....	4
1.1.3 A választott fejlesztési eszközök.....	4
1.2 A weboldal elérhetővé tétele a világhálón.....	5
2. Frontend dokumentáció.....	6
2.1 Nyitó oldal.....	6
2.2 Az autók listázása.....	7
2.3 Az autó lefoglalása.....	8
2. 4 Ügyfél bejelentkezés.....	9
2.5 Ügyfél regisztráció.....	10
3. Adatbázis tervezés.....	11
4. A backend oldalon alkalmazott megoldások.....	14
4.1 A controllerek és endpointok.....	15
4.1.1 Booking controller.....	15
5. Tesztelési dokumentáció.....	19
5.1 Bejelentkezés (login form) tesztelése.....	19
5.2 Az autó foglalás (booking form) tesztelése.....	20
6. Fejlesztési lehetőségek.....	22

## Bevezetés

Egy autójavító cég bővíteni szeretné vállalkozását autóbérlés szolgáltatással. Így növelhetné bevételeit illetve ügyfeleit jobban a céghez köthetné azáltal, hogy akár sürgős, akár előre eltervezett esetben autót tud biztosítani a számukra.

A kényelem és a könnyű elérhetőség miatt célszerű egy webes bérlési lehetőséget biztosítani az ügyfeleknek.

Az autóbérlő weboldalaknak számos előnye van, amelyek miatt sokan használják őket autók bérlésére. Néhány ok, amiért szükség lehet egy ilyen weboldalra:

Kényelmes és egyszerű: Az autóbérlő weboldalak segítségével könnyedén és gyorsan tudsz autót bérelni otthonról, az interneten keresztül. Nincs szükség személyes megjelenésre az autókölcsönzőnél, amely időt takarít meg és kényelmes lehet.

Nagy választék: Az autóbérlő weboldalak általában nagy választékot kínálnak autókból, így könnyen megtalálhatod az igényeidnek megfelelő autót. Ezen kívül általában kategóriák szerint rendezik a járműveket, így könnyen megtalálhatod az általad keresett típust és méretet is.

Jobb árak: Az autóbérlő weboldalaknak általában jobb árai vannak, mint az autókölcsönzőknél. Az online platformok általában akciókat és kedvezményeket kínálnak, amelyek segítségével spórolhatsz a bérlésen.

Adatbiztonság: Az autóbérlő weboldalak általában biztonságos és megbízható fizetési lehetőségeket kínálnak, amelyekkel biztonságosan tudsz autót bérelni. Ezen kívül a weboldalak általában magas szintű adatvédelemmel rendelkeznek, így biztonságban tudhatod a személyes adataidat is.

Összességében az autóbérlő weboldalak számos előnyt kínálnak az autókölcsönzés során, amelyek miatt sokan preferálják őket. Kényelmesek, nagy választékot kínálnak, jobb árakat biztosítanak, rugalmasak és biztonságosak.

# 1. Felhasználói dokumentáció

## 1. 1 Felhasználói követelmények

### 1.1.1 Általános elvárások

Alap elvárás, hogy a rendszer könnyen hozzáférhető legyen, de csak az arra jogosultak számára.

A kezelő felület átlátható és egyszerű legyen azok számára is, akik nem rendelkeznek komoly informatikai tudással.

### 1.1.2 Specifikus elvárások

A vállalat elsősorban külföldi ügyfelei számára szeretné elérhetővé tenni az autókölcsönzést, ezért az oldal angol nyelven készült.

Az autókról képeket kell megjeleníteni és azok alapadatait is elérhetővé kell tenni a a döntés megkönnyítése érdekében.

A kölcsönzésen kívül vállalat más szolgáltatásait is meg kell jeleníteni.

A weboldalnak reszponzivnak kell lennie, hogy különböző eszközökön is megjeleníthető legyen.

### 1.1.3 A választott fejlesztési eszközök

A backend oldali programozás megvalósításához a Java SE 15 és a Spring Boot 2.7.1 verziókat használtam. A Spring Boot-hoz a Gradle 7.1 verziójú projektépítő eszközt vettem igénybe.

A Java nyelv objektum orientáltsága és platform függetlensége miatt jól alkalmazható az ilyen alkalmazások elkészítéséhez. A Java Spring Boot egy Java keretrendszer, amely segítségével gyorsan és hatékonyan lehet webalkalmazásokat készíteni Java nyelven. A kódoláshoz az IntelliJ IDEA Community Edition 2020.2.3 fejlesztő eszközt használtam.

A frontend oldal megjelenítéséhez a React.js keretrendszert valamint CSS-t illetve Bootstrap könyvtárakat használtam. Ezek a technológiák biztosítják, hogy a weboldal reszponzív legyen.

A webfejlesztéshez a Visual Studio Code-ot használtam.

Az adatbázist a POSTGRESQL adatbázis kezelő-rendszerrel hoztam létre.

## **1.2 A weboldal elérhetővé tétele a világhálón**

Egy weboldal megjelenítéséhez és elérhetővé tételéhez szükség van egy számítógépes szerverre, amelyen a weboldal fájljai és adatbázisai tárolódnak.

Ehhez a cégnek tárhelyet kell bérelnie egy megfelelő szolgáltatótól.

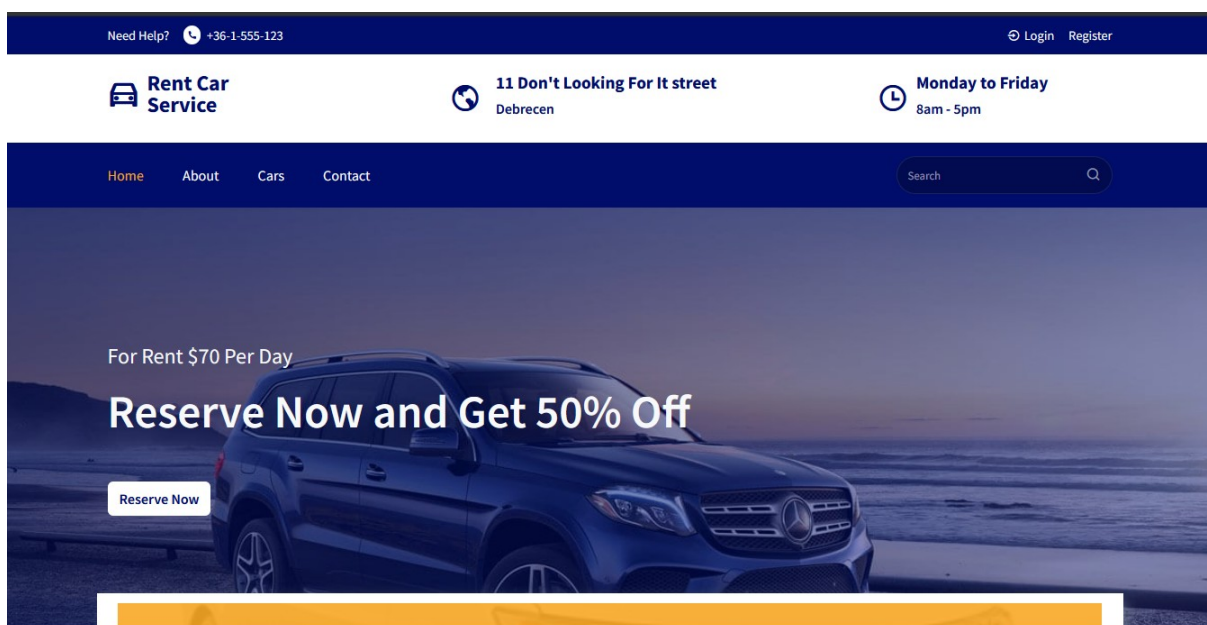
A weboldal megjelenítéséhez szükséges egy domain név is, amely azonosítja a weboldalt a világhálón. A domain név a weboldal elérhetőségét biztosítja a felhasználók számára.

Fontos megemlíteni, hogy a weboldal megjelenítése, a tárhely és a domain név szolgáltatásokat különböző szolgáltatók kínálják. Ezek a szolgáltatók általában havi vagy éves díjat számítanak fel a szolgáltatásokért. Ezeket a költséget a vállalkozásnak kell állnia.

## 2. Frontend dokumentáció

### 2.1 Nyitó oldal

A Home.jsx egy React komponens, amely a honlap főoldalát definiálja. A komponens importál néhány React és Bootstrap komponenst, mint például a Container, Row és Col, amelyek segítségével strukturálja az oldal elrendezését és elhelyezi rajta a különböző alkomponenseket.



A komponens több alkomponenst is tartalmaz, például a HeroSlider-t, amely egy diavetítőt jelenít meg a képekkel, valamint egy űrlapot, amely segítségével a felhasználók autókat kereshetnek az oldalon. A komponens tartalmaz egy "About" részt, amely bemutatja a vállalatot, valamint egy "Services" részt, amely bemutatja az általuk nyújtott szolgáltatásokat.

A komponensnek van egy "Car offer" része is, amely egy autót jelenít meg, és lehetővé teszi a felhasználók számára, hogy a megfelelő ajánlatot találják az oldalon.

A komponens a Helmet komponenst is használja, amely segítségével a weboldal címét állíthatjuk be. Ez az információ segíthet a keresőmotoroknak és a

felhasználóknak megtalálni az oldalt, és az oldal címe jelenik meg a böngésző lapfülén is.


## Find your best car here

Find Car

### About Us

#### Welcome to car rent service

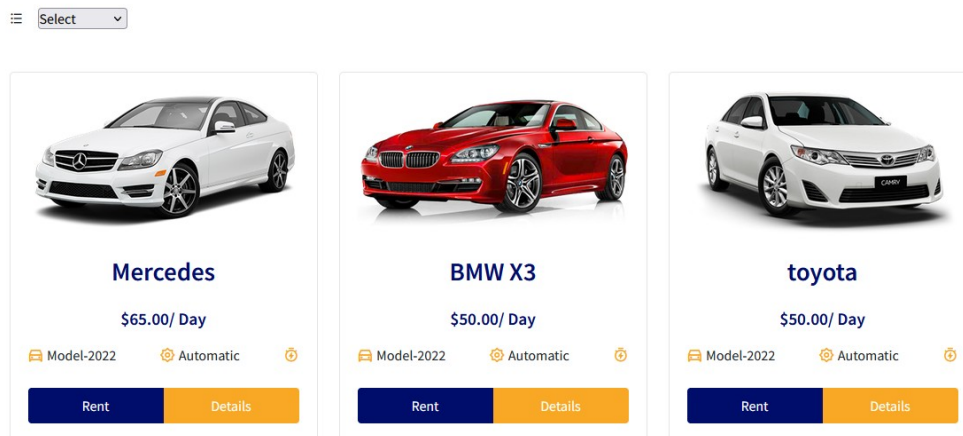
Lorem ipsum dolor sit amet consectetur, adipisicing elit. Voluptatum blanditiis esse accusantium dignissimos labore laborum. Veniam, corporis mollitia temporibus, in quaeat vero deleniti amet dolorem repudiandae, pariatur nam dolore! Impedit neque sit ad temporibus quam similique dolor ipsam praesentium sunt.



## 2.2 Az autók listázása

A CarListing.jsx az autók listáját jeleníti meg a weboldalon, ami egyik legfontosabb eleme a frontend résznek, mivel a leendő ügyfelek itt tájékozódhatnak a bérelhető autókról. Az autókat lehet ár szerint növekvő vagy csökkenő sorrendbe rendezni a lenyíló menü segítségével. A ReactJS könyvtáron kívül a Bootstrap-ot és CSS-t használok az oldal formázásához.

A useState hookot használja az állapotok kezeléséhez, és az useEffect hookot arra, hogy megváltoztassa az ikont a sorrend szerinti rendezésre történő váltáskor. A CommonSection és a CarItem komponensek felhasználásával jeleníti meg az oldalt. A kód a Helmet modult is használja a weboldal fejrészének beállításához.



## 2.3 Az autó lefoglalása

Megjeleníti az autó adatait, és lehetővé teszi a felhasználó számára, hogy lefoglalja azt egy foglalási űrlap segítségével.

A CarDetails komponens importál React hookokat, például useState és useEffect, valamint egyéb összetevőket, például Container, Row, Col, Helmet, useParams és BookingForm-ot.

A useParams hook segítségével kinyerhető a slug paraméter az URL-ből.

A useState a cars és carssingleCarItem állapotváltozók beállítására szolgál.

A useEffect segítségével lekérhető az összes autóadatot egy külső szolgáltatásból a getAllCars-ból importált funkció segítségével ../services/GetCarsData.js. Ezután beállítja a slug paraméternek megfelelő singleCarItem állapotváltozóját.

Végül az importált BookingForm lehetővé teszi a felhasználó számára a kiválasztott autó lefoglalását.





Dolor labore lorem no accusam sit justo sadipsing labore invidunt voluptua, amet duo et gubergren vero gubergren dolor. At diam. Dolor labore lorem no accusam sit justo

Model-2022 Automatic speed200  
GPS Navigation seat-yes Mercedes

#### Booking Information

<input type="text" value="First name"/>	<input type="text" value="Last Name"/>
<input type="text" value="Email"/>	<input type="text" value="Phone Number"/>
<input type="text" value="From Address"/>	<input type="text" value="To Address"/>
<input type="text" value="éééé . hh . nn ."/>	<input type="text" value="-- : --"/>
<input type="text" value="éééé . hh . nn ."/>	<input type="text" value="-- : --"/>

## 2. 4 Ügyfél bejelentkezés

A LoginPage componens egy bejelentkezési űrlapot jelenít meg. Az űrlap bekéri a felhasználó e-mail-címét és jelszavát, POST request-et (kérést) küld a szerver végpontjának, és sikeres hitelesítés esetén átirányítja a felhasználót a profiloldalára. Az összetevő a React Bootstrap könyvtárat használja az űrlapelemek megjelenítéséhez, és más függőségeket is importál, például a useNavigateReact Router DOM-ból, és Popuphibaüzeneteket jelenít meg.

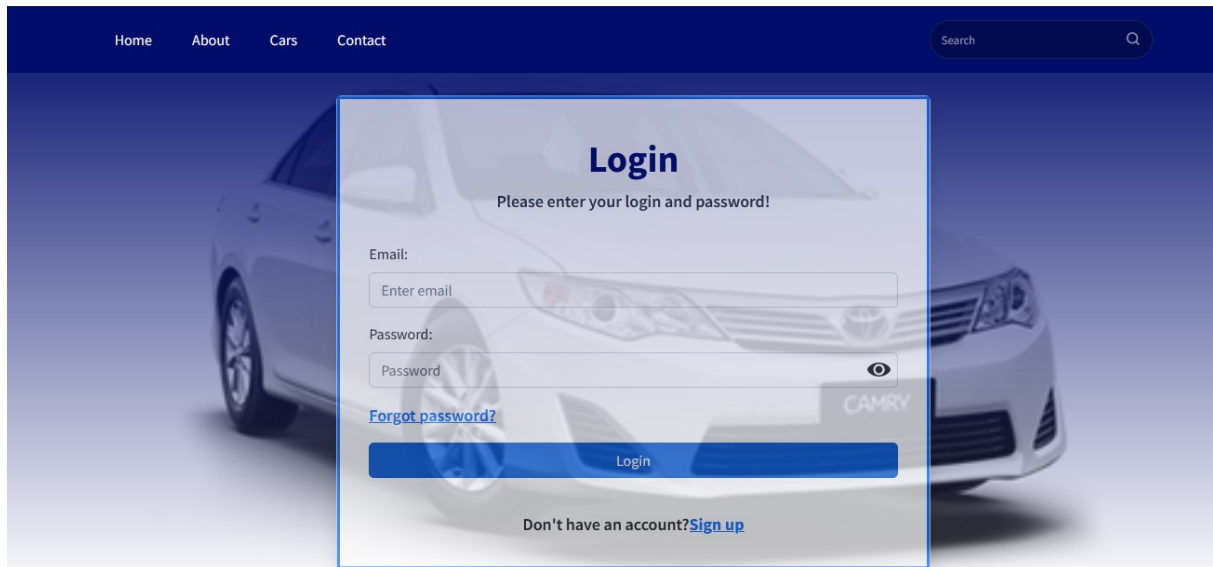
A komponens több állapotváltozót definiál a useState segítségével. Ezek közé tartozik a felhasználó e-mailje, jelszava, egy logikai érték, amely jelzi, hogy a jelszó látható vagy rejtett, egy logikai érték, amely jelzi, hogy megjelenjen-e a hibaüzenet előugró ablaka, maga a hibaüzenet, egy logikai érték, amely azt jelzi, hogy a felhasználó be van-e jelentkezve vagy sem, és végül, a helyi tárhelyen tárolt érték annak ellenőrzésére, hogy a felhasználó már bejelentkezett-e.

A handlePopup függvény beállítja, hogy az üzenet megjelenjen a felugró ablakban. A handleClosePopup függvény showPopup false értékre áll be, ha a felugró ablak bezárul.

A handleSubmit függvény kezeli az űrlap beküldését. POST Kérést küld a szervernek a felhasználó által megadott e-mail címmel és jelszóval. Ha a kérés sikeres

(állapotkód 200), az állapotot igazra állítja isLoggedIn, az értéket a helyi tárhelyen tárolja, és a felhasználót a profiloldalára irányítja a useNavigate segítségével. Ha a kérés sikertelen, akkor az popupMessageállapotot a szerver által visszaadott hibaüzenetre 'error' állítja be .popupErrorMessage.

Végül, ha az autentikáció sikeres, akkor az ügyfél saját profil oldalára érkezik.

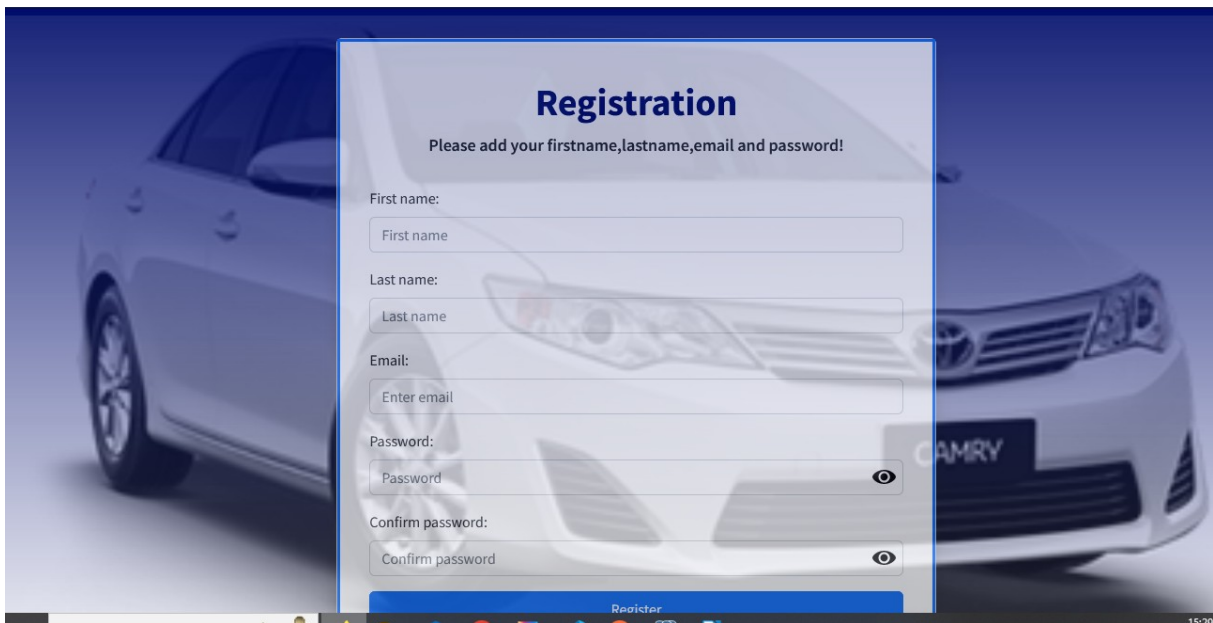


## 2.5 Ügyfél regisztráció

Ez egy regisztrációs oldal, amely biztosítja az új ügyfél adatainak felvitelét. Az oldal tartalmaz egy űrlapot, amely a felhasználói adatokat, például keresztnévet, vezetéknévet, e-mail-címet, jelszót és a jelszó megerősítését tartalmazza. Tartalmaz egy felugró komponenst is, amely üzeneteket és figyelmeztetéseket jelenít meg a felhasználó számára.

A validatePassword függvény meghívása annak ellenőrzésére, hogy a megadott jelszó és a megerősítő jelszó egyezik-e vagy sem. Ha a jelszavak egyeznek, a felugró ablakban megjelenik a „sikeres” állapotú üzenet. Ha a jelszavak nem egyeznek, a felugró ablakban megjelenik egy „hiba” állapotú üzenet.

A `handlePopup` függvény az előugró üzenet beállítására szolgál, a `handleClosePopup` funkció pedig a popup komponens elrejtésére szolgál. A `handleSubmit` függvény akkor kerül meghívásra, amikor a felhasználó elküldi az űrlapot. POST kérést küld a szervernek a felhasználó adataival, és ha a válasz állapota 200, a felhasználó adatai törlődnek, és a felugró ablakban egy sikerüzenet jelenik meg. Ellenkező esetben hibaüzenet jelenik meg a felugró ablakban, és a kiszolgáló hibaüzenete naplózásra kerül a konzolon.

A registration form titled "Registration" is displayed as a modal overlay on a background image of a white Toyota Camry. The form has a light blue header with the title and a subtitle "Please add your firstname,lastname,email and password!". It contains five input fields: "First name:", "Last name:", "Email:", "Password:", and "Confirm password:". Each field has a placeholder text. The "Password:" and "Confirm password:" fields have toggle icons (an eye) to show or hide the password. At the bottom of the form is a blue button labeled "Register". The background image shows the front of the car, with the word "CAMRY" visible on the license plate area. The time "15:29" is visible in the bottom right corner of the image.

### 3. Adatbázis tervezés

Az adatbázist a programozás során figyelembe veendő objektumokhoz terveztem. Az adatbázist az autóbérlés angol elnevezése alapján hoztam létre, "carrental" néven. A létrehozott adatbázis öt táblát tartalmaz, amelyek neve "booking", "car\_image", "cars", "customers" és "employee", valamint öt szekvencia, amelyek ezekhez a táblákhoz elsődleges kulcsként szolgálnak.

A "booking" tábla információkat tartalmaz az autófoglalásokról, mint például az átvétel dátuma, a leadás dátuma, az átvétel időpontja, a leadás időpontja, az átvétel

címe és a leadás címe, valamint megjegyzéseket arra az esetre, ha az ügyfélnek valami kérése, közlendője van..

A "car\_image" tábla az autó és a képazonosítók közötti kapcsolat tárolására szolgál.

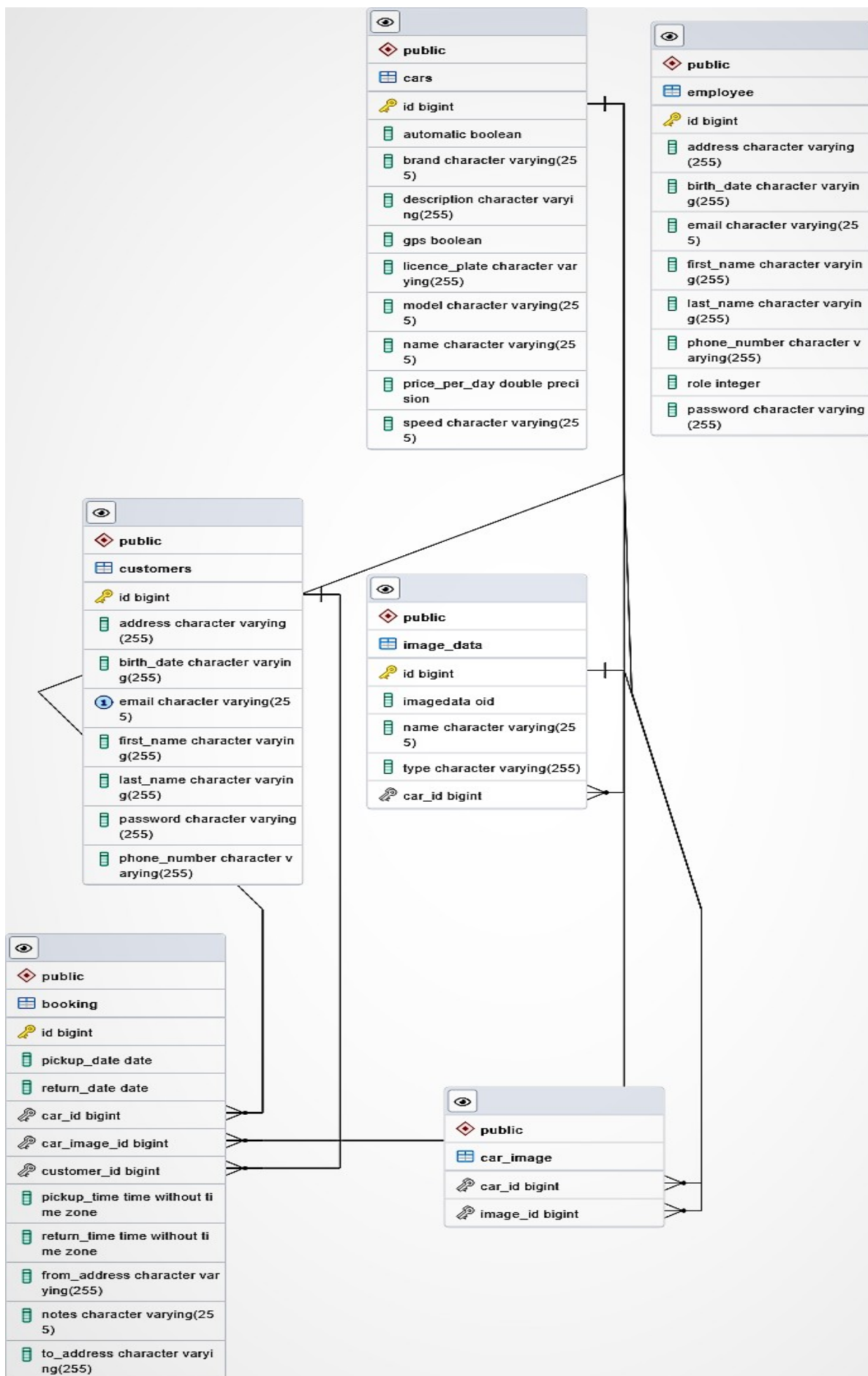
Az "cars" táblázat információkat tartalmaz az autókról, beleértve a márkát, modellt, rendszámot, napi árat, valamint azt, hogy van-e GPS és automata sebességváltó.

Az „customers” táblázat az ügyfelek adatait tárolja. A nevét, címét, e-mail címét, telefonszámát és jelszavát.

Az „employee” táblázat az alkalmazottak adatait tárolja, beleértve a nevét, címét, e-mail címét, telefonszámát, szerepkörét és jelszavát.

Végül az "image\_data" tábla az autókhoz kapcsolódó képeket tárolja nevükkel, típusukkal és képadatokkal együtt.

Ez alapján az adatbázis a következőképpen épül fel:



## 4. A backend oldalon alkalmazott megoldások

Az alábbiakban részletesen mutatom be a különböző feladatokhoz választott megoldásokat. Ezekből jól látható, hogy Java egyedi kódokat használtam.

A Java Spring Boot- ban az MVC elveit alkalmaztam. Az MVC (Model-View-Controller) egy olyan tervezési minta, amely az alkalmazást három egymással összefüggő összetevőre osztja: Modell, View és Controller.

Az MVC-hez alkalmazkodva 4 package-t hoztam létre. Ezek a controllers, models, repositories, services.

**Modellek:** Ezek azok az osztályok, amelyek az alkalmazás adatstruktúráit képviselik. Általában olyan mezőket tartalmaznak, amelyek a rendszerben lévő entitások attribútumaira vannak leképezve, valamint az adatokon az alapvető CRUD (Create, Read, Update, Delete) műveletek végrehajtására szolgáló módszerek. A Spring Boot rendszerben ezeket az osztályokat @Entity annotációval látják el, mert egy ORM (Object-Relational Mapping) keretrendszer, például a Hibernate felügyeli és kezeli.

**Vezérlők:** Ezek azok az osztályok, amelyek kezelik az ügyfelek HTTP-kéréseit, és interakcióba lépnek a modellekkel és szolgáltatásokkal a válaszok generálásához. Jellemzően olyan metódusokat tartalmaznak, amelyek meghatározott URL-ekhez és HTTP-metódusokhoz vannak hozzárendelve, és képesek kezelni az adatok érvényesítését, hitelesítését és engedélyezését is. A Spring Boot rendszerben ezek az osztályok is annotációval vannak ellátva, mint pl. @RestController és a HTTP metódusok meghatározásához olyan annotációkat használnak, mint a @GetMapping, @PostMapping, stb.

**Repozitorik:** (ezek magyarul raktárat lerakatot jelentenek, de a fejlesztésben megmaradt az angolos elnevezés): Ezek azok az interfészek, amelyek meghatározzák az adatbázissal való interakció műveleteit. Általában kibővítik a Spring Data JPA által biztosított CrudRepository vagy JpaRepositoryi nterfészeket, amelyek az alapvető CRUD-műveleteket biztosítják. Egyéni metódusok is

meghatározhatók a Spring Data JPA elnevezési konvencióinak követésével vagy a `@Query` megjegyzés használatával.

**Services:**(ebben az esetben is van magyar megfelelője, a szolgáltatások, de az angol elnevezést használjuk). Ezek azok az osztályok, amelyek magukba foglalják az alkalmazás üzleti logikáját. Általában olyan metódusokat tartalmaznak, amelyek összetett műveleteket hajtanak végre egy vagy több modell és adattár használatával. A Spring Boot rendszerben ezek az osztályok gyakran vannak `@Service` annotációval ellátva, és használhatják a Spring's Dependency Injection-t (DI) függőségeik kezelésére.

Összességében ezek az összetevők együttműködve egy jól szervezett, méretezhető és karbantartható kódbázist hoznak létre a Spring Boot alkalmazáshoz.

## 4.1 A controllerek és endpointok

Az MVC rendszerben a controller biztosítja az endpointokat és a frontend ezeken keresztül kapja meg a kért adatokat, ezért az endpointokat, API-kat emelem ki a backend fejlesztésből. Hiszen ide már a modellen, repositorin, servicen keresztül jutnak el a kért adatok.

A webfejlesztésben az endpoint egy adott URL-re vagy URI-ra utal, amely válaszol a HTTP-kérésekre. Ez egy módja annak, hogy egy kliens, például egy webböngésző vagy egy mobilalkalmazás kommunikáljon a szerverrel, és kérjen bizonyos funkciókat vagy adatokat.

### 4.1.1 Booking controller

Ez egy Java Spring Boot vezérlőosztály, melynek neve `BookingController`. Különbféle végpontokat tartalmaz az entitáshoz kapcsolódó HTTP-kérelmek kezelésére `Booking`.

A `@RestController` annotáció jelzi, hogy ez egy vezérlőosztály, és `@RequestMapping("/booking")` a vezérlő összes végpontjának az URL-útvonalhoz való leképezésére szolgál `/booking`.

Az `@Autowired` annotáció a controllerbe beilleszti `BookingRepository` és `BookingService` osztályok egy példányát.

Az osztály végpontjai a következők:

`@GetMapping("/get/all")`: Ez a végpont az összes foglalás listáját adja vissza a `BookingRepository findAllBy()` metódusával.

`@GetMapping("/get/id/{id}")`: Ez a végpont egyetlen foglalást ad vissza a foglalás id azonosítója alapján a `BookingRepository findById()` metódusával

`@PostMapping("/save")`: Ez a végpont új foglalást hoz létre a `BookingService`. `SaveOrUpdateABooking()` metódusával

`@PutMapping("/{id}")`: Ez a végpont frissít egy meglévő foglalást a `BookingRepository findById()` és `save()` metódusaival

`@DeleteMapping("delete/{id}")`: Ez a végpont töröl egy meglévő foglalást a foglalás id-je és a `BookingRepository findById()` és `delete()` metódusainak felhasználásával.

Ezenkívül használom `@CrossOrigin` annotációt hogy lehetővé tegye a keresztirányú kéréseket bármely tartományból. Az `@RequiredArgsConstructor` annotáció segítségével automatikusan létrejön egy konstruktor az összes szükséges függőséggel.

#### **4.1.2 Car controller**

Ez a `CarController` osztály, ami felelős az alkalmazás Car modelljéhez kapcsolódó HTTP kérések kezeléséért.

A vezérlő számos végpontot biztosít, amelyek segítségével CRUD (Create, Read, Update, Delete) műveleteket lehet végrehajtani a modellen Car.

Az annotációk megegyeznek vagy hasonlóak a `BookingController` osztálynál leírtakkal. Ezért ezt nem részletezem.



A controller az alábbi endpointokat biztosítja::

POST /car/save: Új autót hoz létre az adatbázisban.

POST /car/{id}/images: Képet tölt fel a megadott azonosítóval rendelkező autóhoz.

GET /car/get/all: Az összes autót lekéri az adatbázisból.

PUT /car/edit/{id}: Frissíti a megadott azonosítóval rendelkező autó adatait.

DELETE /car/delete/{id}: Törli a megadott azonosítóval rendelkező autót az adatbázisból.

A controller, a korábbiakhoz hasonlóan a Lombok-könyvtárat is használja olyan mintakódok generálására, mint például a konstruktor és a getterek/setterek injektálása.

#### **4.1.3 Car controller**

A CustomerController osztály a következő metódusokat és végpontokat tartalmazza:

registerCustomer: Ezzel a metódussal új ügyfelet regisztrálunk a rendszerbe. Először ellenőrzi, hogy az ügyfél firstName és lastName létezik-e már az adatbázisban. Ha igen, akkor BAD\_REQUEST üzenettel válaszol. Ezután ellenőrzi, hogy az ügyfél e-mail-címe létezik-e már az adatbázisban, és ha igen, akkor BAD\_REQUEST üzenettel válaszol. Ezután validálja az ügyfél jelszavát a CustomerService osztály isValidPassword metódusával. Ha a jelszó érvénytelen, akkor BAD\_REQUEST üzenettel válaszol. Végül létrehoz egy új Customer objektumot a szükséges adatokkal, kódolja a jelszót a PasswordEncoder segítségével, és elmenti az új ügyfelet az adatbázisba és OK üzenettel válaszol.

authenticateUser: Ezzel a metódussal e-mail-lel és jelszóval hitelesítheti az ügyfeleket. Az AuthenticationManager-t használja az ügyfél hitelesítésére, beállítja a hitelesítést a SecurityContextHolder-ban, és OK üzenetet küld vissza. Ha a hitelesítés sikertelen, UNAUTHORIZED üzenettel válaszol.

logoutUser: Ezzel a módszerrel egy ügyfél kijelentkezett. Az aktuális hitelesítést a SecurityContextHolder,-tól kapja meg és ha nem null, akkor a SecurityContextLogoutHandler OK választ ad.

getCustomerByEmail: Ezzel a módszerrel az ügyfelek e-mail címeik alapján kereshetők. Paraméterként egy e-mail címet vesz fel, és egy Customer objektumot ad vissza a megfelelő e-mailhez. Ha az ügyfél nem található, akkor InstanceNotFoundException kivételt dob.

#### **4.1.4 Image controller**

Az ImageController nevű REST API vezérlő kezeli a képadatokkal kapcsolatos HTTP-kéréseket, és a Spring Data JPA tárolóin keresztül kapcsolatba lép az adatbázisokkal.

A használt metódusok:

getImageInfoByName(): Információkat kér le egy képről név szerint, amelyet elérési útváltozóként ad át. A metódus meghívja az ImageDataService getImageInfoByName() metódusát egy ImageData objektum lekéréséhez, amelyet a válasz törzsében ad vissza.

getImageByName(): Név szerint kér le egy képet, amelyet elérési útváltozóként ad át. A metódus meghívja az ImageDataService getImage() metódusát, hogy lekérje a képet reprezentáló bájtömböt, amely a választörzsben „image/png” tartalomtípussal kerül visszaadásra.

getAllImagesInByte(): Az adatbázisban ImageData objektumként tárolt összes képet lekéri, amelyek egy listában kerülnek visszaadásra.

deleteByName(): Név szerint törli a képet, amely elérési útváltozóként kerül átadásra. A metódus meghívja az ImageDataRepository deleteByName() metódusát, hogy törölje a képet az adatbázisból.

`deleteByld()`: Töröl egy képet azonosító alapján, amely elérési útváltozóként kerül átadásra. A metódus meghívja az `ImageDataRepository deleteByld()` metódusát, hogy törölje a képet az adatbázisból.

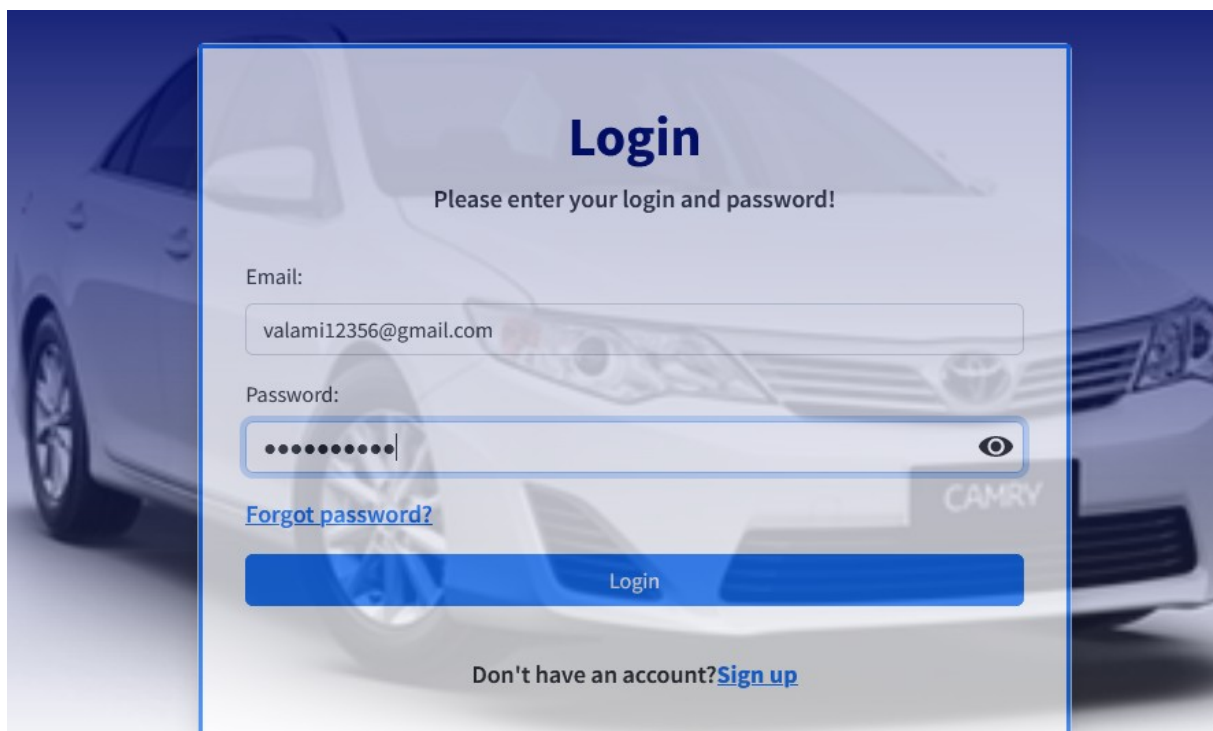
`getImageByld()`: Lekér egy képet azonosító alapján, amely elérési útváltozóként kerül átadásra. A metódus meghívja az `ImageDataRepository findByld()` metódusát egy `ImageData` objektum lekéréséhez, amelyet a válasz törzsében ad vissza.

A `@CrossOrigin` annotáció lehetővé teszi a források közötti erőforrás-megosztást a vezérlő számára, lehetővé téve, hogy a különböző tartományokból származó ügyfelek hozzáférjenek.

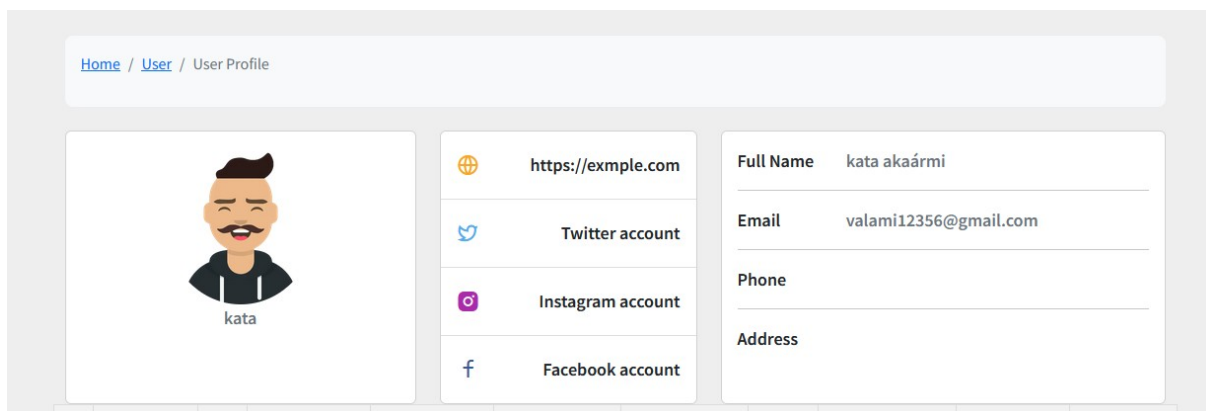
## 5. Tesztelési dokumentáció

### 5.1 Bejelentkezés (login form) tesztelése

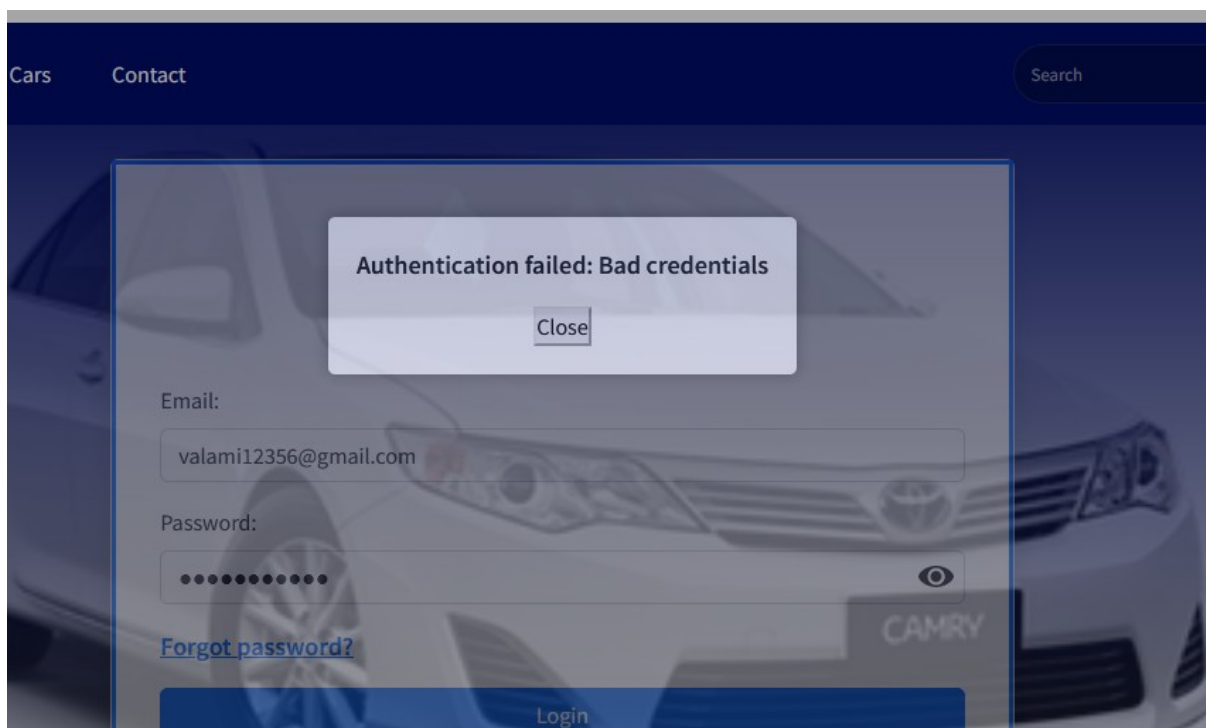
A bejelentkező ablakban beírtam a felhasználói azonosítót és a jelszót. Működött a beléptetés.



A helyes jelszó és email cím megadása után a felhasználó oldalra irányít a program.



A jelszó elgépelés vagy nem megfelelő jelszó esetén hibaüzenetet kapunk.



## 5.2 Az autó foglalás (booking form) tesztelése

A foglalás elkészültekor, a Booking gomb (button) használatával az adatbázisba kerülnek a foglalási adatok. Sikeres foglalás esetén üzenetet kapunk a foglalásról. beírásakor viszont azt a hibát találtam, hogy a beírt új jelszó és a megerősítésként

## Booking Information

kata

akaármi

valami12356@gmail.com

Phone Number

Debrecen

Budapes

2023 . 05 . 31 .

2023 . 06 . 15 .

The booking is successful

Close

Write

Booking

## 6. Fejlesztési lehetőségek

Tisztában vagyok vele, hogy a program több területen is tovább fejleszthető.

Például be lehetne építeni több adatellenőrzést a programba. Most is van adatellenőrzés, de ezt tovább lehetne fejleszteni.

Az autójavító cégnél igény van arra, hogy a cég alkalmazottja figyelemmel kísérje, kövesse a foglalásokat és adatokat kérjen le. Ebből kiindulva az alábbi fejlesztési elképzelésekkel lehetne bővíteni ezt a nyilvántartó rendszert.

- Ki lehetne bővíteni egy alkalmazotti hozzáféréssel. Ehhez már elő van készítve a megfelelő osztály, amiből az Employee objektumk létrehozhatók. munkaidő nyilvántartására alkalmas modullal. Ebből következően az adatbázis tábla is létre van hozva.
- A cég alkalmazott számára statisztikákat lehetne készíteni hogy mikor melyik autót használták a legtöbbet. Melyik volt a legtöbb alkalommal javításon, stb.
- Az ügyfelek részére biztosítani kell a bank kártyas fizetési lehetőséget, ezért egy bankkártyás modullal is ki kell egészíteni a programot. .
- A biztonság növelése érdekében az autentikációs műveletbe több biztonsági modult lehetne beépíteni. Továbbá a jelszó tárolását is biztonságosabbá kell tenni.
- Az ügyfelek aktívabbá tétele miatt lehetne blog modult létrehozni, ahol az ügyfelek vagy az alkalmazottak az autóbérléssel kapcsolatos tapasztalataikat írhatnák le. Ezzel a cég azt is elérné, hogy minél többször keressék fel a weboldalt. Ezáltal még ismertebbé válna.