Békéscsabai SZC Nemes Tihamér Technikum és Kollégium

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Szakképesítés megnevezése: Szoftverfejlesztő és -tesztelő*

*Azonosító száma: 5 0613 12 03*

**VIZSGAREMEK**

Autókölcsönző

Készítették:

Tanuló1: Kocsis Márk Osztály: 2-14FE-Szoft-2223 Oktatási azonosító: 73231995409

Tanuló2: Varga Zoltán Osztály: 2-14FE-Szoft-2223 Oktatási azonosító: 72289980857

**Békéscsaba, 2023/2024**

**NYILATKOZAT**

Alulírott ...................................................................... , a Békéscsabai SZC Nemes Tihamér Technikum és Kollégium tanulója kijelentem, hogy a(z) ...................................................................................................................................................... ...................................................................................................................................................... című vizsgaremek elkészítésében való közreműködésem a saját, önálló munkám, az abban általam hivatkozott nyomtatott és elektronikus szakirodalom felhasználása a szerzői jogok szabályainak megfelelően történt.

Tudomásul veszem, hogy a vizsgaremek esetén plágiumnak számít:

• ha a feladat elkészítője bármilyen más forrásból vett következtetést vagy megoldást a sajátjaként tüntet fel,

• ha a dokumentációk szövegét vagy annak egy részét nem a dolgozaton szerzőként feltüntetett személy írja,

• ha a más forrásból nyert forráskódok, programelemek, adatok, ábrák valamint szövegrészek dolgozatban való felhasználása esetén a forrás megjelölése elmarad vagy a hivatkozás alapján nem azonosítható egyértelműen a forrás.

Alulírott kijelentem, hogy a plágium fogalmát megismertem és tudomásul veszem, hogy plágium esetén a vizsgaremek vizsgarész visszautasításra kerül.

Békéscsaba, 2024. év .............................hó .........nap

…………………………………… ……………………………………

Kocsis Márk Varga Zoltán

# 1. Bevezetés

Az internet megjelenésével az ember nagyon sok új szolgáltatás könnyen és gyorsan tudnak keresni illetve igénybe venni. Az online autókölcsönző szolgáltatásokat az emberek számos kényelmi és praktikus okból használják:

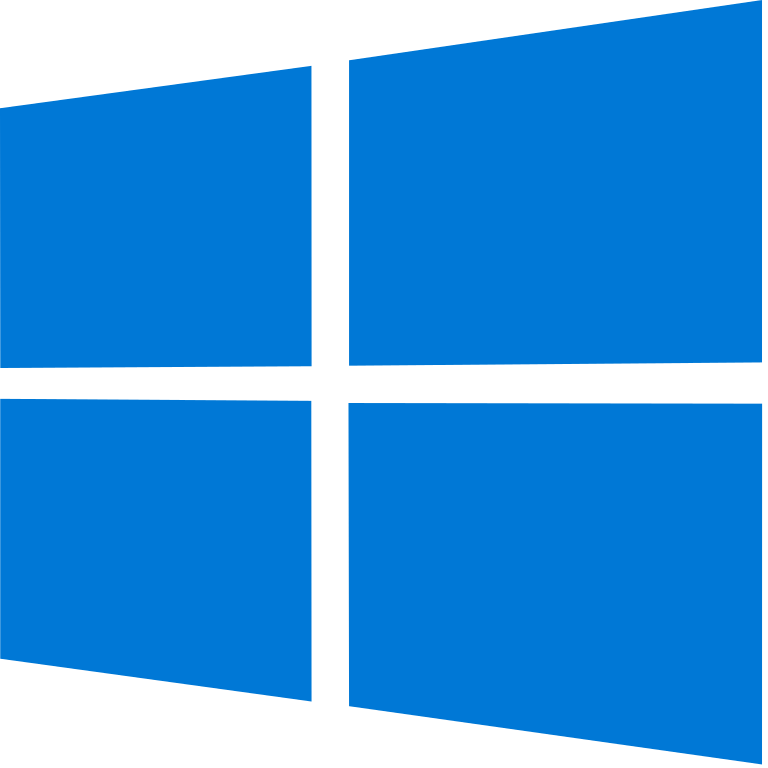
* ***Kényelem***: Az online autókölcsönző platformok lehetővé teszik az autók bérlését a saját otthonukból vagy bármely internetkapcsolattal rendelkező helyről. Nem szükséges személyesen felkeresni egy kölcsönző irodát vagy hosszú sorban állni a bérlésért. Tabletről vagy telefonról is lehet intézni az autó kölcsönzését.
* ***Költséghatékonyság:*** Az online platformok gyakran kínálnak versenyképes árakat és különféle ajánlatokat, így az ügyfelek könnyebben találhatnak megfelelő árral rendelkező autót a számukra megfelelő időszakra.
* ***Könnyű összehasonlítás***: Az online autókölcsönző oldalak lehetővé teszik az autók, árak és feltételek könnyű összehasonlítását. Ez lehetővé teszi az ügyfelek számára, hogy gyorsan megtalálják az igényeiknek és a pénztárcájuknak legjobban megfelelő lehetőséget.
* ***Széles választék:*** Az online autókölcsönző platformok általában széles választékot kínálnak különböző autótípusokból és márkákból..
* ***Folyamatos elérhetőség:*** Az online autókölcsönző szolgáltatások 24/7 elérhetők, így az ügyfelek bármikor foglalhatnak autót, akár hétvégeken vagy ünnepnapokon is.
* ***Egyszerű foglalási folyamat:*** Az online foglalási folyamat általában gyors és egyszerű, és csak néhány kattintást igényel az autó kiválasztásához, az időpontok megadásához és a fizetéshez.

Ezek az előnyök teszik vonzóvá az online autókölcsönző szolgáltatásokat azok számára, akik autót bérelnének, legyen szó üzleti vagy személyes célú utazásról.

A vizsgaremek elkészítésekor az első lépések között volt, hogy létrehozzunk egy weboldalt, amin a felhasználó regisztrálást, belépést követően, könnyedén elkészítheti a foglalását. Illetve egy adatbázis létrehozása, amiben az oldalon beírt adatokat, információkat tárolhatom a rendelés sikeressége érdekében. Mindkettőnknek ez volt az első weboldala, amit létrehoztunk. Elkészítés során nagyon sok új információt és ismeretet szereztünk.

# 2. Fejlesztői dokumentáció

## 2.1. Fejlesztői környezet

****A webalkalmazásunk elkészítése során az alábbi környezetben és programokkal dolgoztunk.

**Windows 10**

A Windows 10 operációs rendszert használtunk vizsgaremek elkészítésénél, mert ez az operációs rendszer kompatibilis a később felsorolt programokkal.



**Visual Studio Code**

A frontend és backend oldalon a *Visual Studio Code 1.89.0*-es verzióját használtuk. Visual Studio Code egy ingyenes, nyílt forráskódú fejlesztői környezet (IDE), amelyet a Microsoft fejlesztett ki. Könnyű, gyors és nagyon testre szabható, így ideális választás fejlesztők számára különböző programozási nyelvekhez és projektekhez. A Visual Studio Code számos funkciót kínál, mint például kódszerkesztés intelligens automatizálással, beépített Git támogatás, bővítmények által kiegészíthető funkcionalitás és integrált hibakereső eszközök. Általánosságban könnyen tanulható, sokféle fejlesztési projekthez használható és népszerű a fejlesztők körében.



**XAMPP**

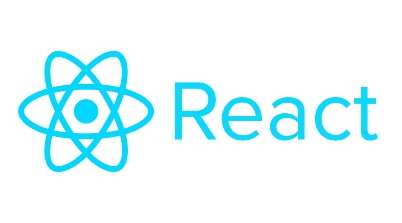
A XAMPP egy ingyenes, nyílt forráskódú szoftvercsomag, amely teljes körű fejlesztési környezetet biztosít web alkalmazások fejlesztéséhez. A név a "Cross-platform (X), Apache, MySQL, PHP, Perl (P)" rövidítéséből ered. A XAMPP tartalmazza az Apache HTTP szervert, a MySQL adatbázis-kezelő rendszert, a PHP-t és a Perl-t, valamint más kiegészítő szoftvereket, például phpMyAdmin-t. Ez a szoftvercsomag segít gyorsan és könnyen telepíteni és konfigurálni egy teljes körű web fejlesztési környezetet a lokális gépen, így ideális választás fejlesztőknek és tanulóknak is.

A XAMPP csomagunk 8.2.12. verziójú, amely az alábbi verziójú szoftvereket tartalmazza:

* *Apache 2.4.58 (Win64)* *webszerver*
* *MariaDB10.4.32* adatbázis-kezelő rendszer
* *PHP 8.2.12* szerveroldali szkriptnyelv
* *phpMyAdmin 5.2.1* egy nyílt forrású eszköz, amit PHP-ban írtak a MySQL menedzselésére az interneten keresztül. Jelenleg képes készíteni és eldobni adatbázisokat, készíteni/eldobni/módosítani táblákat, törölni/módosítani/hozzáadni mezőket, SQL parancsokat futtatni és a mezőkön kulcsokat kezelni.

**POSTMAN**

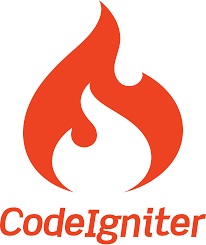
Postman egy népszerű, ingyenes API-fejlesztő platform, amely lehetővé teszi fejlesztőknek, hogy könnyen teszteljék, dokumentálják és hibakeressék az API-kat. A Postman segítségével könnyen lehet HTTP kéréseket küldeni és fogadni, illetve azokat szervezni, és vizsgálni a válaszokat. Ez egy intuitív felhasználói felülettel rendelkezik, valamint tartalmazza az automatizálás lehetőségét is a tesztek futtatásához és az API-k monitorozásához.

**React keretrendszer**

A frontend rész React Javascript-, Tailwind CSS keretrendszerekkel, és Vite build eszközzel került elkészítésre. React egy népszerű JavaScript keretrendszer, amelyet a Facebook fejlesztett ki. Ez egy deklaratív, komponensalapú felhasználói interfész-keretrendszer, amely lehetővé teszi a fejlesztők számára a dinamikus és hatékony webalkalmazások készítését. A React segítségével könnyen létrehozhatunk újrafelhasználható UI komponenseket, amelyeket könnyen lehet kezelni és karbantartani. A JSX (JavaScript XML) segítségével a React lehetővé teszi a HTML és JavaScript kombinálását, ami hatékony és kényelmes módot nyújt a felhasználói felületek létrehozására. A React egyik kulcsfontosságú jellemzője a virtuális DOM, ami lehetővé teszi az alkalmazások gyorsabb és hatékonyabb frissítését és renderelését. Összességében a React egy erőteljes eszköz a modern webalkalmazások fejlesztéséhez, amelyet a fejlesztők nagy hatékonysággal tudnak használni.

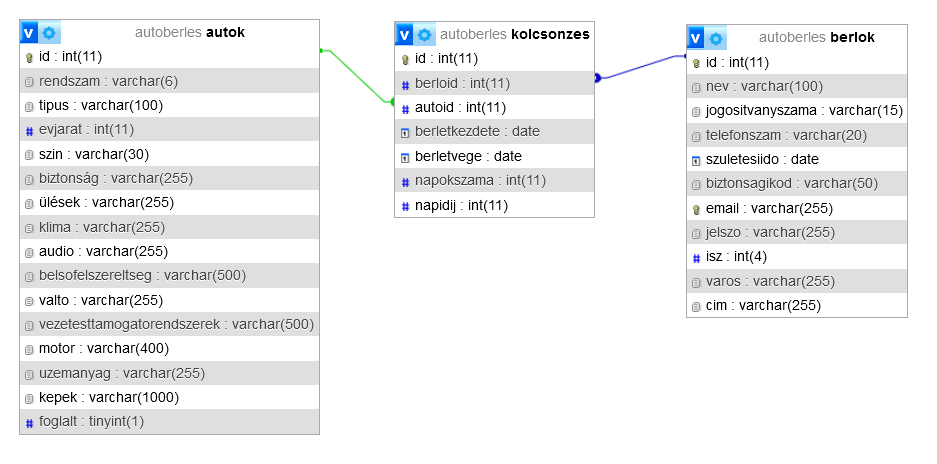
**Tailwind CSS** 

A *Tailwind CSS* egy újító és népszerű CSS keretrendszer, amely lehetővé teszi a fejlesztők számára, hogy gyorsan és hatékonyan építsenek fel modern, testreszabott webes felhasználói felületeket. A Tailwind CSS nem egy hagyományos előre definiált osztályokra épülő keretrendszer, hanem egy utility-first megközelítést alkalmaz. Ez azt jelenti, hogy a fejlesztők egy sor alapvető, alacsony szintű stílusosztályt (pl. margin, padding, színek stb.) használhatnak, amelyeket közvetlenül a HTML elemekhez rendelhetnek. Ez a flexibilitás és testreszabhatóság lehetővé teszi a Tailwind CSS gyors alkalmazását és a stílusok könnyű kezelését.

***CodeIgniter 4***

A backend oldal *CodeIgniter 4* keretrendszerrel lettkialakítva, amely egy népszerű webalkalmazás keretrendszer, Támogatja a komponens alapú fejlesztést, a modern PHP verziókat, és könnyen testre szabható a fejlesztők igényei szerint. A CodeIgniter 4 lehetővé teszi a gyors és hatékony webalkalmazások fejlesztését a PHP világában.

## 2.2. Az adatbázis bemutatása

Az ügyfelek adatainak tárolására illetve a foglalások tárolására az autoberles adatbázist hoztuk létre. Az adatbázisunk UTF-8 (Unicode Transformation Format - 8-bit) karakterkódolású, ami támogatja a magyar nyelvben használt karaktereket, beleértve az ékezetes betűket is Az adatbázis létrehozásakor ki kellett választani, a magyar nyelvet is. Így a magyar nyelvű szövegek kezelésére optimalizált, a magyar nyelvi szabályoknak megfelelően kezeli a kis- és nagybetűk közötti különbségeket, és az ékezetes karaktereket is megfelelően sorrendbe állítja.

1. ábra: Az Autoberles adatbázis táblái, mezői és kapcsolatai.

Forrás:http://localhost/phpmyadmin/index.php?route=/database/designer&db=autoberles

### 

2. ábra: Az Autoberles adatbázis autó típus táblái.

Forrás:http://localhost/phpmyadmin/index.php?route=/database/designer&db=autoberles

### 2.1.1. A berlok tábla

Ez a tábla a rendszer fő adminisztrációs táblája. Itt tároljuk az oldalra regisztrált felhasználók regisztrációját és adatait. Az egyes mezők a következő adatokat tartalmazzák:

* *id* INT(11)). A felhasználók egyedi azonosító száma (PRIMARY KEY). Egész szám típusú (integer) érték. A mező AUTO INCREMENT, azaz minden új felhasználó regisztráció rögzítése során automatikusan növekszik. Az egyik legfontosabb azonosítő. Hiszen ez alapján lehet lekérni,
* *nev* VARCHAR(100). A felhasználó neve. Karakterlánc típusú (varchar), maximum 100 karakter hosszúságú.
* *jogositvanyszama* VARCHAR(15). A felhasználó jogosítványának száma.
* *telefonszam* VARCHAR(20). A felhasználó telefonszámának tárolására szolgál.
* *szuletesiido* DATE. A felhasználó születési idejét menti el dátum formátumban.
* *biztonsagikod* VARCHAR(50) Jelentősége akkor van ha a felhasználó elfelejtette a jelszavát. Ha ezt a kódot pontosan megadja akkor tudja csak módosítani a jelszavát.
* *email:* VARCHAR(255) A regisztráció elején ez az egyik adat, amit meg kell adni. Egy e-mail címmel csak egy felhasználó regisztrálhat. Nem lehet módosítani.
* *jelszo:* VARCHAR(255) A regisztráció elején ez a másik adat, amit meg kell adni. Az adatbázisba titkosítva kerül. Módosítható.
* *isz:* INT(4) A felhasználó címének irányítószáma..
* *város:* VARCHAR(255) A felhasználó címének városa,
* *cim:* VARCHAR(255) A felhasználó címének utca és házszámát tárolja el

### 2.1.2. A kolcsonzes tábla

Az autó foglalásának műveleteit tároló tábla, ami a következő mezőket tartalmazza

* *id* INT(11). A foglalások egyedi azonosító száma (PRIMARY KEY). A mező AUTO INCREMENT, azaz új foglalás rögzítése során automatikusan növekvő számsor.
* *berloid* INT(11). Kapcsolat a berlok táblához, a foglalást tevő felhasználó egyedi azonosítóját tartalmazza.
* *autoid* INT(11). Kapcsolat az autók táblához. A lefoglalt autó egyedi azonosítóját tartalmazza.
* *berletkezdete* (DATE). A foglalás kezdődátuma. Dátum típusú mező.
* *berletvege* (DATE). A foglalás vége dátuma. Dátum típusú mező.
* *napokszama* INT(11): Számított mező. A berletvege és a berletkezdete-ének a különbsége. Nem lehet negatív
* *napidij.* INT(11): A foglalás díja. A napok számával számol*.*

### 2.1.3. Az autok tábla és a merci, kia, tesla, ford, audi táblák

A foglalás során az autok táblázatból választhatók a lefoglalni kívánt autók. A márkaneves táblák felépítése hasonló az autok táblával. Az autó fontosabb adatait tartalmazza. A legfontosabb mezők a következők:

* *id* INT(11). Az autók egyedi azonosító száma (PRIMARY KEY). A mező AUTO INCREMENT, azaz automatikusan növekvő számsor. Kapcsolatban áll a Kolcsonzes táblával.
* *rendszam* VARCHAR(6). Az autó rendszáma
* *tipus* VARCHAR(100). Az autó tipusa. pl: Audi RS e-tron GT
* *evjarat* INT(11) Az autó gyártási éve
* *szin* VARCHAR(30) Az autó színe
* *biztonság* VARCHAR(255) Az autó biztonsági felszereléseit tartalmazza
* *ülések* VARCHAR(255) Az autóban milyen típusú üléseket lehet találni.
* *klima* VARCHAR(255) Az autóban milyen típusú klimát szereltek
* *audio* VARCHAR(255) Az autóban milyen érintőképes rendszer van
* *belsofelszereltseg* VARCHAR(500) Az autóban milyen egyéb kisegítő felszereltség van beszerelve
* *valto* VARCHAR(255) Az autóban milyen váltó van szerelve pl: Egy sebességű automatikus váltó
* *vezetesttamogatorendszerek VARCHAR(500)* Az autóban milyen vezetés támogató rendszerek vannak pl.: Adaptív sebességtartó automatika, Parkolóasszisztens, 360 fokos kamera
* *motor* VARCHAR(400) Az autó lóerejéről és nyomatékáról van adat rögzítve .
* *uzemanyag* VARCHAR(255) Az autó milyen üzemanyagot használ. Benzin, Dizel, Elektoromos.
* *kepek* VARCHAR(1000) Az autóról egy kép van elmentve amit egy URL link mutat meg.

### 2.1.4. Adatbázis kapcsolatok

* Az autok tábla id mezője kapcsolódik a kolcsonzes tábla autoid mezőjéhez.
* A berlok tábla id mezője kapcsolódik a kolcsonzes tábla berloid mezőjéhez.

## 2.2. Az alkalmazás felépítése

Az alkalmazásunk két fő részre osztható frontend részre és backend részre.

A Frontend, a kliens oldali rész. Egyszerűen, amit a felhasználó lát, érzékel, és amivel kapcsolatban tud kerülni. A weboldalak megjelenítésétől, a felhasználói felületeken keresztül (UI), felhasználói élményt (UX) meghatározó elemekig.

A backend a szerver oldali rész. Ez az alkalmazás „agya” Ez a rész kezeli az adatbázis-műveleteket, felhasználói hitelesítéseket, adatok kezelését és tárolását. Felhasználói kéréseket dolgoz fel stb. A backend fejlesztők feladati közé tartozik az API-k kialakítása, amelyeken keresztül a frontend adatokat kérhet.

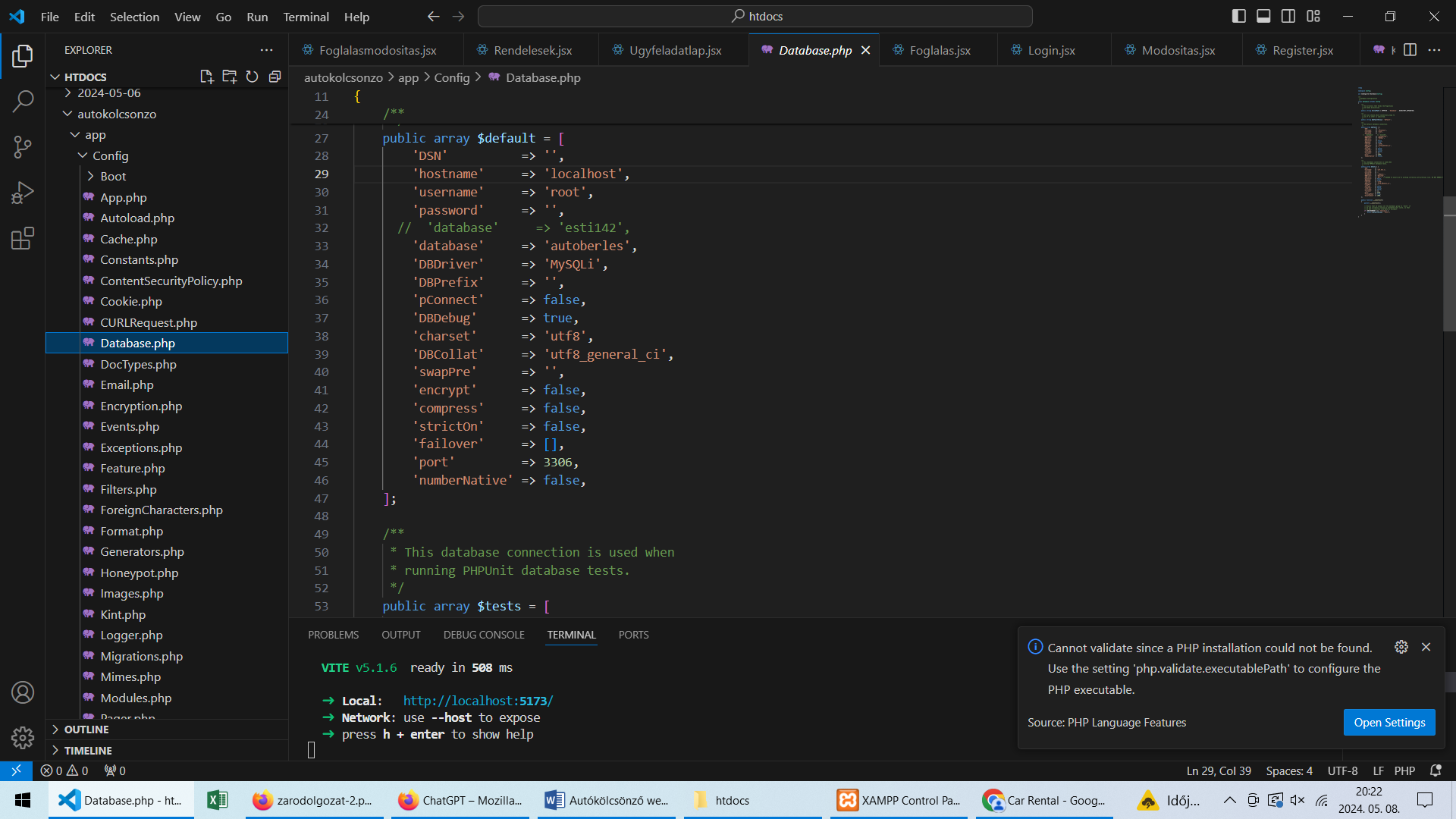
### 2.2.1. Backend

A Backend oldal kialakításához CodeIgniter 4 keretrendszert és MySQL relációs adatbázis-kezelő rendszert használtunk. A korábbi pontban részletezett táblák létrehozása után lehetett elkezdeni és a végpontokat létrehozni. A CodeIgniter 4 egy PHP alapú, könnyű, és erőteljes MVC keretrendszer, amelyet a webalkalmazások és az API-k fejlesztésére terveztek. A CodeIgniter az egyik legnépszerűbb keretrendszer a PHP közösségben. Az egyik jellemzője a MVC Architektúra. A Model-View-Controller architektúra lehetővé teszi az alkalmazás különféle rétegeinek elválasztását.

**Modell (Model)**: A modell réteg az adatkezeléssel és az adatbázishoz való kapcsolattal foglalkozik. CodeIgniterben az adatbázis-műveletek könnyen elvégezhetők az aktív rekordok segítségével.

**Nézet (View)**: A nézet réteg a felhasználói felület kialakításáért felelős. A CodeIgniterben a nézeteket egyszerű PHP fájlokban lehet elkészíteni. Mi erre a frontendet használtuk.

**Vezérlő (Controller):** A vezérlő réteg irányítja a kéréseket és a válaszokat. A CodeIgniterben a vezérlőket osztályokként lehet megvalósítani.

A kezdetben be kellett állítani néhány dolgot. Létre kellett hozni egy autokolcsonzo mappát a XAMPP mappa htdocs mappájában és telepíteni egy üres CodeIgniter rendszert. Be kellett állítani az app/Config mappa Database.php fájlban néhány dolgot hogy működhessenek majd a lekérések.

Meg kellett adni:

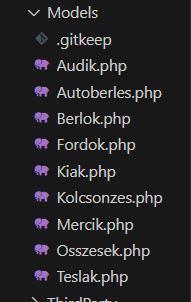
* hostname: localhost
* username: root
* password: ’’
* database: adatbázis nevét
* charset: adatbázis kódolása

3. ábra: A database.php fájl tartalma

A BackEnd programozáskor a két legfontosabb mappa a beállítások után a Models és a Controllers mappák.

*Az adatlekérdezések a modellekből a kontrollerekbe:* A kontrollerek felelősek a kérések fogadásáért, feldolgozásáért és válaszadásáért. Amikor a kontroller szükségét érzi az adatok lekérdezésének vagy módosításának, az általában a modelleket használja fel ehhez. Például, ha egy kontroller egy felhasználó bejelentkezési folyamatával foglalkozik, akkor lekérheti a felhasználó adatait egy felhasználó model segítségével. (Berlok.php model)

*Az adatok feldolgozása a modellekben:* A modellek felelősek az adatok lekérdezéséért és módosításáért az adatbázisból. Ezek az osztályok tartalmazzák az adatbáziskezelő metódusokat, például az adatok lekérdezését, beszúrását, frissítését vagy törlését. Ezek a metódusokat a kontrollerek használják fel az adatok kezelésére.

A Models mappa tartalma:

Az Audik.php a Fordok.php, Kiak, php, Mercik.php, Osszesek.php, Teslak.php modellek az csak lekérdezéseket tartalmaznak. A funkciói lehetővé teszik az összes adat lekérdezését az adott márka táblából.

Az Autoberles.php egy olyan modell, amely az "autok" táblával foglalkozik, és két fő metódust kínál: az összes rekord lekérését és egy adott rekord lekérését az azonosítója alapján.

A Berlok.php modell felelős az adatbázis-műveletek végrehajtásáért és az adatok manipulálásáért a "berlok" adatbázistáblában. A funkciói segítségével kezelhetjük a felhasználók regisztrációját, bejelentkezését, adatmódosítását és egyéb adminisztratív tevékenységeket egy autókölcsönző alkalmazásban.

4. ábra: Modellek

A Kolcsonzes.php felelős az autók kölcsönzésével kapcsolatos adatbázisműveletek végrehajtásáért, beleértve az adatok lekérdezését, beszúrását, frissítését és törlését is.

Nézzük meg részletesen ezeket a Modelleket.

### 2.2.1.1. Berlok modell

<?php

namespace App\Models;

use CodeIgniter\Model;

class Berlok extends Model

{

    protected $table ="berlok";

    protected $returnType = "object";

   protected $allowedFields=["id", "nev", "jogositvanyszama", "telefonszam", "szuletesiido","jelszo","email","isz","varos","cim", "biztonsagikod" ];//update-nél van jelentősége

public function getAll() //lista() lista2()

    {

        return $this->findAll(); // mindent visszaad

    }

    public function getLoginId($id)//Navbarnál kéri le a belépett felhasználó adatait

    { //listaid() listaidmod()

        return $this->where("id", $id)->findAll(); // mindent visszaad

    }

    public function insAdat($adat)

    { //register()

        return $this->insert($adat, false); //adatbeszúrás a alaphelyzetben a visszatérési értéke true lesz és a tid et adja vissza

        //Ha a beírjuk hogy false akkor tudunk változtatni rajta.

    }

    public function vanemail($email)

    { //register()

        $vizsgal=$this->where('email', $email)->findAll();

        if (count($vizsgal)>0) //  ha nagyobb, mint 0 akkor azt jelenti van és talált

            return true;

        else

            return false;

    }

    public function van($email, $jelszo)

    { //belepes()

        $vizsgal=$this->where('email', $email)->where('jelszo', $jelszo)->findAll();

        if (count($vizsgal)>0) //  ha nagyobb mint 0 akkor, azt jelenti van talált

        return true;

        else

        return false;

    }

    public function vanbizkod($email, $biztonsagikod)

    { //biztonsagi()

        $vizsgal=$this->where('email', $email)->where('biztonsagikod', $biztonsagikod)->findAll();

        if (count($vizsgal)>0) //  ha nagyobb mint 0 akkor azt jelenti van talált

        return true;

        else

        return false;

    }

    public function updAdat($id, $modadat){

        //jelszomod() adatmodosit()

        $this->update($id, $modadat);

    }

    public function getVar($email)

    {  //biztonsagi()

        $vizsgal=$this->where('email', $email)->findAll();

        if (count($vizsgal)>0) //  ha nagyobb mint 0 akkor azt jelenti van talált

            return $vizsgal[0]->id;

        else

            return null;

    }

    public function getId($id)

    { //Nem lett használva

        $vizsgal=$this->where('id', $id)->findAll();

        if (count($vizsgal)>0) //  ha nagyobb mint 0 akkor azt jelenti van talált

            return $vizsgal;

        else

            return null;

    }

}

5. ábra Berlok.php modell tartalma

Az elején történik az adatbázistábla és tulajdonságok definiálása. A $table, $returnType és $allowedFields tulajdonságok meghatározzák a modell alapvető beállításait. A $table a használt adatbázistábla nevét adja meg, a $returnType az adatok visszaadási típusát határozza meg, (object) míg az $allowedFields a modell számára engedélyezett mezőket határozza meg az adatbázistáblából. A berlok tábla összes mező nevét felsoroltuk.

Adatok lekérdezése: Az getAll() metódus az összes adat lekérdezéséért felelős a "berlok" táblából. A findAll() metódust használja az adatok lekérdezéséhez, amely az összes rekordot visszaadja.

Belépési adatok ellenőrzése: A van($email, $jelszo) metódus ellenőrzi, hogy a megadott e-mail cím és jelszó páros megtalálható-e a "berlok" táblában. Ha a megadott kombináció megtalálható, a metódus true értéket ad vissza, egyébként false értéket. A belépésnél van használatban.

Biztonsági kód ellenőrzése: A vanbizkod($email, $biztonsagikod) metódus ellenőrzi, hogy a megadott e-mail címhez tartozó biztonsági kód megtalálható-e a "berlok" táblában. A jelszó módosításnál van jelentősége.

Adat frissítése: Az updAdat($id, $modadat) metódus frissíti az adatokat a megadott azonosítójú rekordban a "berlok" táblában. Két helyen használja a kontroller. A jelszó módosításnál és a felhasználó adatainak módosításánál.

Azonosító lekérése e-mail cím alapján: A getVar($email) metódus lekéri az adott e-mail címhez tartozó azonosítót a "berlok" táblából. A jelszó módosításnál használjuk.

Azonosító alapján lekérdezés: A getId($id) metódus lehetővé teszi az azonosító alapján történő lekérdezést, bár jelenleg nincs használatban a kódban.

### 2.2.1.2. Kolcsonzes modell

<?php

namespace App\Models;

use CodeIgniter\Model;

class Kolcsonzes extends Model

{

    protected $table ="kolcsonzes";

    protected $returnType = "object";

    protected $allowedFields=["id", "berloid", "autoid", "berletkezdete", "berletvege", "napokszama", "napidij"];//updatenál van jelentősége

    public function getAll()

    { //lista()

        return $this->findAll(); // mindent visszad

    }

    public function rendeles($berloid)

    {   //rendelesek(), rendelesekid()

        return $this->select('kolcsonzes.berloid, kolcsonzes.id, berlok.nev, autok.tipus, kolcsonzes.autoid, kolcsonzes.berletkezdete, kolcsonzes.berletvege, kolcsonzes.napokszama, kolcsonzes.napidij, autok.kepek')

                    ->join('berlok', 'kolcsonzes.berloid=berlok.id')

                    ->join('autok', 'kolcsonzes.autoid=autok.id')

                    ->where('berlok.id', $berloid)->findAll();

    }

    public function insBerles($adat)

    {   //ujadatbeszur()

        return $this->insert($adat, false); //adatbeszúrás a alaphelyzetben a visszatérési értéke true lesz és a tid et adja vissza

        //Ha a beírjuk hogy false akkor tudunk változtatni rajta.

    }

    public function vankolcson($autoid) // megvizsgálja, hogy az autót kikölcsönöztzék e?

    {   //kolcsonmod()

        $vizsgal=$this->where("autoid", $autoid)->findAll(); // mindent visszad

        if (count($vizsgal)>0) //  ha nagyobb mint 0 akkor azt jelenti van talált

        return $vizsgal;

        else

        return false;

    }

    public function delkolcsonzes($id)

    {   //kolcsontorolid()

        $this->where("id", $id)->delete(); // kölcsönzés idre megy a törlés

    }

    public function updAdat($id, $modadat){ //kolcsonmod()

        $this->update($id, $modadat);

    }

    public function getId($id)//lekéri a rendelés főbb adatait

    {   //nem használjuk

        return $this->where("id", $id)->findAll(); // mindent visszad

    }

    public function rendelesid($rendid)

    {   //rendelesekid()

        return $this->select('kolcsonzes.berloid, kolcsonzes.id, berlok.nev, autok.tipus, autok.id, kolcsonzes.berletkezdete, kolcsonzes.berletvege, kolcsonzes.napokszama, kolcsonzes.napidij, autok.kepek')

                    ->join('berlok', 'kolcsonzes.berloid=berlok.id')

                    ->join('autok', 'kolcsonzes.autoid=autok.id')

                    ->where('kolcsonzes.id', $rendid)->findAll(); }

}

6. ábra Kolcsonzes.php modell tartalma

A "App\Models\Kolcsonzes" névtérrel van ellátva. A modell a "kolcsonzes" adatbázistáblával dolgozik, és különböző adatbázisműveleteket végez a kölcsönzések kezelése során. Itt is az elején történik az adatbázistábla és tulajdonságok definiálása.

Adatok lekérdezése: Az getAll() metódus az összes kölcsönzési adat lekérdezéséért felelős a "kolcsonzes" táblából. A findAll() metódust használja az adatok lekérdezéséhez, amely az összes rekordot visszaadja.

Kölcsönzések lekérése berlő azonosító alapján: A rendeles($berloid) metódus lekéri az adott berlőhöz tartozó kölcsönzéseket, és visszaadja azokat egyesítve az "autok" és "berlok" táblák adataival.

Kölcsönzés beszúrása: Az insBerles($adat) metódus új kölcsönzési adatokat szúr be a táblába.

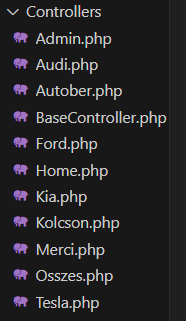
Kölcsönzés meglétének ellenőrzése autó azonosító alapján: A vankolcson($autoid) metódus ellenőrzi, hogy egy adott autóra már létezik-e kölcsönzés a táblában.

Kölcsönzés törlése: A delkolcsonzes($id) metódus törli a megadott azonosítójú kölcsönzést a táblából.

Adatok frissítése: Az updAdat($id, $modadat) metódus frissíti az adatokat a megadott azonosítójú rekordban a táblában.

Azonosító alapján lekérdezés: A getId($id) metódus lehetővé teszi az azonosító alapján történő lekérdezést a táblából.

Rendelés azonosító alapján lekérdezése: A rendelesid($rendid) metódus lekéri az adott rendelés azonosítóhoz tartozó kölcsönzéseket, és visszaadja azokat egyesítve az "autok" és "berlok" táblák adataival.

Ez a modell a kölcsönzési adatok kezelésére szolgál, és lehetővé teszi azok lekérdezését, beszúrását, frissítését és törlését a "kolcsonzes" táblában.

### 2.2.2. Kontrollerek

Az Audi.php, Autober.php, Fork.php, Kia.php, Merci.php, Osszes.php, Tesla.php kontrollerek a hasonlo márka nevű modelleket használják, csak lekérdezéseket tartalmaznak. Mind a BaseController osztályból származik, ami azt jelenti, hogy örökli a CodeIgniter alapvető kontroller funkcióit és metódusait. A lista metódus egy központi műveletet definiál a kontrollerben. Ez a metódus felelős az adatok lekérdezéséért és visszaadásáért az ügyfél felé. A metódus kezeli a HTTP kérést, és ha az a GET metódussal érkezik, akkor lekéri az adatokat a modell segítségével, majd JSON formátumban visszaadja azokat a válaszként. Ha a kérés nem GET metódussal érkezik, akkor egy 400-as státuszkóddal és egy megfelelő üzenettel tér vissza.

7. ábra Kontrollerek

### 2.2.2.1. Admin kontroller

Az Admin kontroller a Berlok modellt használva számos funkciót biztosít az adminisztrációs felülethez, mint például felhasználók kezelése, bejelentkezés, adatok módosítása. Mindegyik metódus először ellenőrzi a kérés típusát, és csak a megfelelő HTTP metódussal érkező kéréseket fogadja el. Amennyiben nem megfelelő a metódus akkor egy 400-as státuszkóddal és egy „Nem megfelelő HTTP metódus” üzenettel tér vissza.

<?php

namespace App\Controllers;

use App\Models\Berlok;

use App\Helpers\JWTHelper;

class Admin extends BaseController

{

    //http://localhost/autokolcsonzo/public/admin/lista

    public function lista()

    {

        $request =request();

        if($request->getMethod()=="get")

        {

            $model=model(Berlok::class);//csatlakozunk a modelhez

            $adatok=$model->getAll();

            $response=response();

            return $response->setJSON($adatok);

        }

        else

        {

            $response=response();

            return $response->setStatusCode(400)->setBody("Nem megfelelő HTTP metódus");

        }

    }

    //http://localhost/autokolcsonzo/public/admin/lista2

    public function lista2()

    {

        $request =request();

        if($request->getMethod()=="get")

        {

            $model=model(Berlok::class);//csatlakozunk a modelhez

            $adatok=$model->getAll();

            $response=response();

            return $response->setJSON($adatok);

        }

        else

        {

            $response=response();

            return $response->setStatusCode(400)->setBody("Nem megfelelő HTTP metódus");

        }

    }

      //http://localhost/autokolcsonzo/public/admin/listaid

      //belépett felhasználó adatait kéri le.

      public function listaid()

      {

          $request=request();

          $bejovo=$request->getJSON();

          $id = $bejovo->listaid;

          if($request->getMethod()=="post")t

          {

            $model=model(Berlok::class);//csatlakozunk a modelhez

            if (isset($id))

            {

                $adatok = $model->getLoginId($id);

                $response=response();

                return $response->setJSON($adatok);

            }

            else

            {

                $response=response();

                return $response->setStatusCode(400)->setBody("Hiányzó paraméter");

            }

         }

          else

          {

              $response=response();

              return $response->setStatusCode(400)->setBody("Nem megfelelő HTTP metódus");

          }

      }

      //http://localhost/autokolcsonzo/public/admin/listaidmod

//belépett felhasználó adatait kéri le a módosításhoz

      public function listaidmod()

      {

          $request=request();

          $bejovo=$request->getJSON();

          $id = $bejovo->listaidmod;

          if($request->getMethod()=="post")//get volt

          {

            $model=model(Berlok::class);//csatlakozunk a modelhez

            if (isset($id))

            {

                $adatok = $model->getLoginId($id);

                $response=response();

                return $response->setJSON($adatok);

            }

            else

            {

                $response=response();

                return $response->setStatusCode(400)->setBody("Hiányzó paraméter");

            }

         }

          else

          {

              $response=response();

              return $response->setStatusCode(400)->setBody("Nem megfelelő HTTP metódus");

          }

      }

    //http://localhost/autokolcsonzo/public/admin/register

    public function register()

    {

        $request=request(); // kérés

        if ($request->getMethod()=="post") // ha a kérés típusa post (adatbevitel)

        {

            $frontendtol=$request->getJSON();

            if(!isset($frontendtol->nev) || !isset($frontendtol->email) || !isset($frontendtol->jelszo))

            {

                $response =response();

                $response->setStatusCode(400);

                $response->setBody("Hiányos adatok."); // a hibaüzenet kiírása

                $response->send();

            }

            else

            {

                $hashedPassword = password\_hash($frontendtol->jelszo, PASSWORD\_DEFAULT);

                $beadat = [

                    'nev' => $frontendtol->nev,

                    'email' => $frontendtol->email,

                    'jelszo' => $hashedPassword,

                ];

                $model=model(Berlok::class);

                $letezoemail=$model->vanemail($beadat['email']);

                if ($letezoemail==false)

                {

                    if ($model->insAdat($beadat)==true) // tartalma és a típusa is megegyezik e

                        {

                        $response=response();

                        return $response->setStatusCode(201)->setJSON(["newid"=>$model->getInsertID()]);

                        }

                        else

                        {

                        $response=response();

                        return $response->setStatusCode(400)->setBody("Sikertelen adatlétrehozás");

                        }

                }

                else

                {

                    $response=response();

                    $response->setStatusCode(400);

                    $response->setBody("Ezzel az e-mail címmel regisztráltak már."); // a hibaüzenet kiírása

                    $response->send();

                }

            }

        }

        else

        {

            $response=response();

            $response->setStatusCode(400);

            $response->setBody("Nem megfelelő Methódus típus"); // a hibaüzenet kiírása

            $response->send();

        }

    }

//http://localhost/autokolcsonzo/public/admin/belepes

    public function belepes()

    {

        $request=request(); // kérés

        if ($request->getMethod()=="post") l)

        {

            $adatok=$request->getJSON();

            if((!isset($adatok->email)) || (!isset($adatok->jelszo)))

            {

                $response =response();

                $response->setStatusCode(400);

                $response->setBody("Hiányos adatok."); // a hibaüzenet kiírása

                $response->send();

            }

            else

            {

                $hashedPassword = password\_verify($adatok->jelszo, PASSWORD\_DEFAULT);

                $beadat = [

                    'email' => $adatok->email,

                    'jelszo' => $hashedPassword,

                ];

                $model=model(Berlok::class);

                $letezofelhasznalo=$model->van($beadat['email'], $beadat['jelszo']);

                if ($letezofelhasznalo==true)

                {

                    $id=$model->getVar($beadat['email']);

                    $response=response();

                    $response->setStatusCode(200);

                    $response->setJSON($id);

                    $response->send();

                }

                else

                {

                    $response=response();

                    $response->setStatusCode(400);

                    $response->setBody("Nincs ilyen adatokkal regisztrált felhasználó !"); // a hibaüzenet kiírása

                    $response->send();

                }

            }

        }

        else

        {

            $response=response();

            $response->setStatusCode(400);

            $response->setBody("Nem megfelelő Methódus típus"); // a hibaüzenet kiírása

            $response->send();

        }

    }

    //http://localhost/autokolcsonzo/public/admin/adatmodosit

    public function adatmodosit()

    {

        $request=request();

        if($request->getMethod()=="post")

        {

            $adatok = $request->getJSON();

            $id = $adatok->id;

            if (isset($id))

            {

            $modadat = [

                'nev'=>$adatok->nev,

                'jogositvanyszama'=>$adatok->jogositvanyszama, //a jsonből érkező változó nem muszáj hogy azonos legyen

                'telefonszam'=>$adatok->telefonszam,

                'szuletesiido'=>$adatok->szuletesiido,

                'isz'=>$adatok->isz,

                'cim'=>$adatok->cim,

                'varos'=>$adatok->varos,

                'biztonsagikod'=>$adatok->biztonsagikod,

            ]; // ez jön a frontendtől amit már asszociatív tömbbé alakítottunk

                $model = model(Berlok::class);

                $model->updAdat($id, $modadat);

                $response=response();

                return $response->setStatusCode(200)->setBody("OK");

            }

            else

            {

                $response = response(); //ez nem kell mert másképpen nem is jutunk ide.

                return $response->setStatusCode(400)->setBody("Hiányzó ID");

            }

        }

        else

        {

            $response = response();

            return $response->setStatusCode(400)->setBody("Rossz methódus");

        }

    }

     //http://localhost/autokolcsonzo/public/admin/biztonsagi

     public function biztonsagi()

     {

         $request=request(); // kérés

         if ($request->getMethod()=="post")

         {

             $adatok=$request->getJSON();

             if((!isset($adatok->email)) || (!isset($adatok->biztonsagikod)))

             {

                 $response =response();

                 $response->setStatusCode(400);

                 $response->setBody("Hiányos adatok."); // a hibaüzenet kiírása

                 $response->send();

             }

             else

             {

                 $beadat = [

                     'email' => $adatok->email,

                     'biztonsagikod' => $adatok->biztonsagikod,

                 ];

                 $model=model(Berlok::class);

                 $letezofelhasznalo=$model->vanbizkod($beadat['email'], $beadat['biztonsagikod']);

                 if ($letezofelhasznalo==true)

                 {

                     $id=$model->getVar($beadat['email']);

                     $response=response();

                     $response->setStatusCode(200);

                     $response->setJSON($id);

                     $response->send();

                 }

                 else

                 {

                     $response=response();

                     $response->setStatusCode(400);

                     $response->setBody("Nincs ilyen adatokkal regisztrált felhasználó !"); // a hibaüzenet kiírása

                     $response->send();

                 }

             }

         }

         else

         {

             $response=response();

             $response->setStatusCode(400);

             $response->setBody("Nem megfelelő Methódus típus"); // a hibaüzenet kiírása

             $response->send();

         }

      }

     //http://localhost/autokolcsonzo/public/admin/jelszomod

    public function jelszomod()

    {

        $request=request();

        if($request->getMethod()=="post")

        {

            $adatok = $request->getJSON();

            $id = $adatok->id;

            if (isset($id))

            {

            $modadat = [

                'jelszo'=>$adatok->jelszo,

            ];

                $model = model(Berlok::class);

                $model->updAdat($id, $modadat);

                $response=response();

                return $response->setStatusCode(200)->setBody("Sikeres módosítás");

            }

            else

            {

                $response = response(); //ez nem kell mert másképpen nem is jutunk ide.

                return $response->setStatusCode(400)->setBody("Hiányzó ID");

            }

        }

        else

        {

            $response = response();

            return $response->setStatusCode(400)->setBody("Rossz methódus");

        }

    }

}

8. ábra: Az Admin kontroller

A kontroller számos különböző végpontot tartalmaz, amelyek mindegyike egy-egy külön funkciót valósít meg.

A lista és lista2 metódusok felelősek a "berlok" tábla minden adatának lekérdezéséért (GET metódus) és JSON formátumban történő visszaadásáért. A lista végpontot a Foglalas komponensnél használjuk.

A listaid metódus a bejelentkezett felhasználó adatainak lekérdezését végzik (POST metódus), felhasználó azonosító alapján. Navbar komponens a végpont alapján kéri le a belépett felhasználó adatait.

A listaidmod metódusok a bejelentkezett felhasználó adatainak lekérdezését végzik (POST metódus), a felhasználó adatainak módosításhoz az Ugyfeladatlap, Foglalas és Modositas komponenseknél.

A register metódus az új felhasználó regisztrációját kezeli. Ellenőrzi a beérkező adatokat, rögzíti azokat az adatbázisban, és visszaadja az újonnan létrehozott felhasználó azonosítóját JSON formátumban. Register komponens ezzel a végponttal rögzíti az új felhasználót.

A belepes metódus a felhasználó bejelentkezését kezeli. Ellenőrzi a beérkező adatokat, összehasonlítja azokat az adatbázisban tárolt jelszóval, majd visszaadja a bejelentkezett felhasználó azonosítóját JSON formátumban. A végpont segítségével lehet belépni a Login komponensben.

A adatmodosit metódus által létrehozott végponttal a Modositas komponens a felhasználó adatainak módosítását végzi. Fogadja a módosítandó adatokat, frissíti azokat az adatbázisban.

A biztonsagi metódus a felhasználó biztonsági kódjának ellenőrzését végzi. Ellenőrzi a beérkező adatokat, összehasonlítja azokat az adatbázisban tárolt kóddal, majd visszaadja a felhasználó azonosítóját JSON formátumban. Új jelszó beállításánál használjuk ezt a végpontot az Emlekezteto komponensben.

A jelszomod metódust szintén az Emlékezteto komponensben használjuk végpontként, ami a felhasználó jelszavának módosítását végzi. Fogadja a módosítandó jelszót, frissíti azt az adatbázisban.

### 2.2.2.2. Kolcson kontroller

A Kolcson kontroller a Kolcsonzes modellt használva az autókölcsönző alkalmazás fő funkcióit valósítják meg, beleértve az adatok listázását, beszúrását, módosítását és törlését is. Mindegyik metódus először ellenőrzi a kérés típusát, és csak a megfelelő HTTP metódussal érkező kéréseket fogadja el. Amennyiben nem megfelelő a metódus akkor egy 400-as státuszkóddal és egy „Nem megfelelő HTTP metódus” üzenettel tér vissza.

<?php

namespace App\Controllers;

use App\Models\Kolcsonzes;

class Kolcson extends BaseController

{

    //http://localhost/autokolcsonzo/public/kolcson/lista

    public function lista()

    {

        $request = request();

        if ( $request->getMethod() == 'get' )

        {

            $model = model( Kolcsonzes::class );

            //csatlakozunk a modelhez

            $adatok = $model->getAll();

            $response = response();

            return $response->setJSON( $adatok );

        } else {

            $response = response();

            return $response->setStatusCode( 400 )->setBody( 'Nem megfelelő HTTP metódus' );

        }

    }

    //http://localhost/autokolcsonzo/public/kolcson/rendelesek

    public function rendelesek()//adott felhasználó rendelései

    {

        $request = request();

        $bejovo = $request->getJSON();

        $id = $bejovo->berloid;

        if ( $request->getMethod() == 'post' )

        {

            $model = model( Kolcsonzes::class );

            //csatlakozunk a modelhez

            $adatok = $model->rendeles( $id );

            $response = response();

            return $response->setJSON( $adatok );

        } else {

            $response = response();

            return $response->setStatusCode( 400 )->setBody( 'Nem megfelelő HTTP metódus' );

        }

    }

    //http://localhost/autokolcsonzo/public/kolcson/ujadatbeszur

    public function ujadatbeszur()

    {

        $request = request();

        // kérés

        if ( $request->getMethod() == 'post' ) // ha a kérés típusa post ( adatbevitel )

        {

            $frontendtol = $request->getJSON();

            $beadat = [

                'berloid'=>$frontendtol->berloid,

                'autoid'=>$frontendtol->autoid, //a jsonből érkező változó nem muszáj hogy azonos legyen

                'berletkezdete'=>$frontendtol->berletkezdete,

                'berletvege'=>$frontendtol->berletvege,

                'napokszama'=>$frontendtol->napokszama,

                'napidij'=>$frontendtol->napidij,

            ];

            $model = model( Kolcsonzes::class );

            if ( $frontendtol->berletkezdete>$frontendtol->berletvege )

            {

                $response = response();

                return $response->setStatusCode( 400 )->setBody( 'A kezdő dátum későbbi mint a vég dátum' );

            } else {

                $van = $model->vankolcson( $beadat[ 'autoid' ] );

                if ( $van == true )

                {

                    $adatbazis\_kezdo = array\_column( $van, 'berletkezdete' );

                    $adatbazis\_vege = array\_column( $van, 'berletvege' );

                    foreach ( $adatbazis\_kezdo as $adatbazis\_kezdo\_datum )

                    {

                        if ( strtotime( $frontendtol->berletkezdete ) > strtotime( $adatbazis\_kezdo\_datum ) )

                        {

                            foreach ( $adatbazis\_vege as $adatbazis\_vege\_datum )

                            {

                                if ( strtotime( $frontendtol->berletvege ) > strtotime( $adatbazis\_vege\_datum ) )

                                {

                                    if ( $model->insBerles( $beadat ) )

                                    {

                                        $response = response();

                                        return $response->setStatusCode( 201 )->setJSON( [ 'newid' => $model->getInsertID() ] );

                                    } else {

                                        $response = response();

                                        return $response->setStatusCode( 400 )->setBody( 'Sikertelen adatlétrehozás' );

                                    }

                                } else {

                                    $response = response();

                                    return $response->setStatusCode( 400 )->setBody( 'Az autót már lefoglalták.' );

                                }

                            }

                        } else {

                            $response = response();

                            return $response->setStatusCode( 400 )->setBody( 'Az autót már lefoglalták.' );

                        }

                    }

                } else {

                    if ( $model->insBerles( $beadat ) == true ) // tartalma és a típusa is megegyezik e

                    {

                        $response = response();

                        return $response->setStatusCode( 201 )->setJSON( [ 'newid'=>$model->getInsertID() ] );

                    } else {

                        $response = response();

                        return $response->setStatusCode( 400 )->setBody( 'Sikertelen adatlétrehozás' );

                    }

                }

            }

        } else {

            $response = response();

            return $response->setStatusCode( 400 )->setBody( 'Nem megfelelő Methódus típus' );

            // ha nem POSt-al küld a frontend adatot

        }

    }

    //http://localhost/autokolcsonzo/public/kolcson/kolcsontorolid

    public function kolcsontorolid()

    {

        $request = request();

        $bejovoadat = $request->getJSON();

        $id = $bejovoadat->id;

        if ( $request -> getMethod() == 'delete' && isset( $id ) )

        {

            $model = model( Kolcsonzes::class );

            $model->delkolcsonzes( $id );

            $response = response();

            return $response->setStatusCode( 200 )->setBody( 'Törlés sikeres: '.$id );

        } else {

            $response = response();

            return $response->setStatusCode( 400 )->setBody( 'Rossz kérés' )->send();

        }

    }

    //http://localhost/autokolcsonzo/public/kolcson/kolcsonmod

    //belépett felhasználó adatait kéri le a módosításhoz

    public function kolcsonmod()

    {

        $request = request();

        if ( $request->getMethod() == 'post' )

        {

            $adatok = $request->getJSON();

            $id = $adatok->id;

            if ( isset( $id ) )

            {

                $modadat = [

                    'id'=>$adatok->id,

                    'berloid'=>$adatok->berloid,

                    'autoid'=>$adatok->autoid,

                    'tipus'=>$adatok->tipus,

                    'berletkezdete'=>$adatok->berletkezdete, //a jsonből érkező változó nem muszáj hogy azonos legyen

                    'berletvege'=>$adatok->berletvege,

                    'napokszama'=>$adatok->napokszama,

                    'napidij'=>$adatok->napidij,

                ];

                $model = model(Kolcsonzes::class);

                if(($adatok->berletkezdete) > ($adatok->berletvege))

                {

                    $response = response();

                    return $response->setStatusCode(400)->setBody('A kezdő dátum későbbi mint a vég dátum');

                }

                else

                {

                $van = $model->vankolcson($modadat[ 'autoid' ] );

                if (count($van) > 0 )

                    {

                        $adatbazis\_kezdo = array\_column( $van, 'berletkezdete' );

                        $adatbazis\_vege = array\_column( $van, 'berletvege' );

                        foreach ( $adatbazis\_kezdo as $adatbazis\_kezdo\_datum )

                        {

                            if ( strtotime( $adatok->berletkezdete ) > strtotime( $adatbazis\_kezdo\_datum ) )

                            {

                                foreach ( $adatbazis\_vege as $adatbazis\_vege\_datum )

                                {

                                    if ( strtotime( $adatok->berletvege ) > strtotime( $adatbazis\_vege\_datum ) )

                                        {

                                        $model->updAdat( $id, $modadat );

                                        $response = response();

                                        return $response->setStatusCode( 200 )->setBody( 'A fogalalást sikeresen módosította' );

                                        }

                                        else

                                        {

                                        $response = response();

                                        return $response->setStatusCode( 400 )->setBody( 'Az autót már lefoglalták.' );

                                        }

                                    }

                                }

                                else

                                {

                                $response = response();

                                return $response->setStatusCode( 400 )->setBody( 'Az autót már lefoglalták.' );

                                }

                        }

                    }

                    else

                    {

                        $model->updAdat( $id, $modadat );

                        $response = response();

                        return $response->setStatusCode( 200 )->setBody( 'OK' );

                    }

                }

            }

            else

            {

                $response = response();

                //ez nem kell mert másképpen nem is jutunk ide.

                return $response->setStatusCode( 400 )->setBody( 'Hiányzó ID' );

            }

        }

        else

        {

            $response = response();

            return $response->setStatusCode( 400 )->setBody( 'Rossz methódus' );

        }

    }

    //http://localhost/autokolcsonzo/public/kolcson/rendelesekid

    public function rendelesekid()//adott rendelés adatainak lekérdezése

    {

        $request = request();

        $bejovo = $request->getJSON();

        $id = $bejovo->rendid;

        if ( $request->getMethod() == 'post' )

        {

            $model = model( Kolcsonzes::class );

            $adatok = $model->rendelesid( $id );

            $response = response();

            return $response->setJSON( $adatok );

        } else {

            $response = response();

            return $response->setStatusCode( 400 )->setBody( 'Nem megfelelő HTTP metódus' );

        }

    }

}

9. ábra: A Kolcson kontroller

A lista metódus felelős az összes kölcsönzés listázásáért.

A rendelesek a metódus adott felhasználó rendeléseit listázza ki. A kérés típusa POST, és a bejövő adatok alapján lekéri az adott felhasználó rendeléseit a modellen keresztül. A végpontot a Rendelesek komponens használja a rendelések megjelenítéséhez.

Az ujadatbeszur metódus végpontját a Foglalas komponens használja ami felelős új kölcsönzési adatok beszúrásáért. A kérés típusa POST, és a bejövő adatok alapján beszúrja az új kölcsönzést a modellen keresztül. Ellenőrzi, hogy az autó szabad-e a megadott időpontban, és hibaüzenetet ad vissza, ha már foglalt az autó.

A kolcsontorolid a metódus felelős egy kölcsönzés törléséért. A kérés típusa DELETE, és a bejövő adatok alapján törli a megadott kölcsönzést a modellen keresztül. Rendelesek komponens használja a végpontot.

A kolcsonmod a metódus felelős egy kölcsönzés módosításáért. A kérés típusa POST, és a bejövő adatok alapján módosítja a megadott kölcsönzést a modellen keresztül. Ellenőrzi, hogy az autó szabad-e a megadott időpontban, és hibaüzenetet ad vissza, ha már foglalt az autó. Foglalasmodositas komponens a végpont segítségével listázza ki az adott foglalás adatait és lehet vele módosítani.

A rendelesekid metódus adott rendelés adatainak lekérdezéséért felelős. A kérés típusa POST, és a bejövő adatok alapján lekéri az adott rendelés adatait a modellen keresztül. A végpontot a Rendelesek komponens használja.

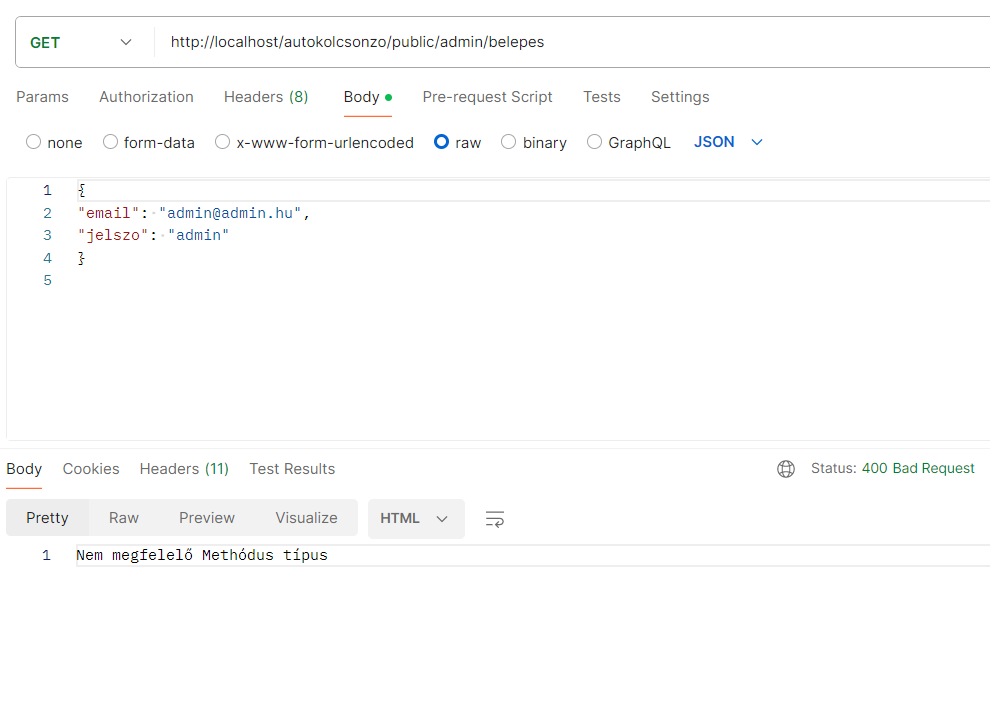
#### 2.2.3. Backend tesztelések

Minden applikáció készítésének egy fontos folyamata a tesztelés. A tesztelést egységekre bontva, mind backend, mind frontend oldalon kell elvégeznünk.

A teszteket a Postman programmal valósítottuk meg, amely széles körben használt alkalmazás, amely lehetővé teszi API-k tesztelését, fejlesztését és dokumentálását Lehetővé teszi, hogy különböző metódusú („POST”,”PUT”, „GET”, „DELETE” stb.) HTTP kéréseket küldjünk végpontokra és annak eredményéről információt kapjunk vissza.

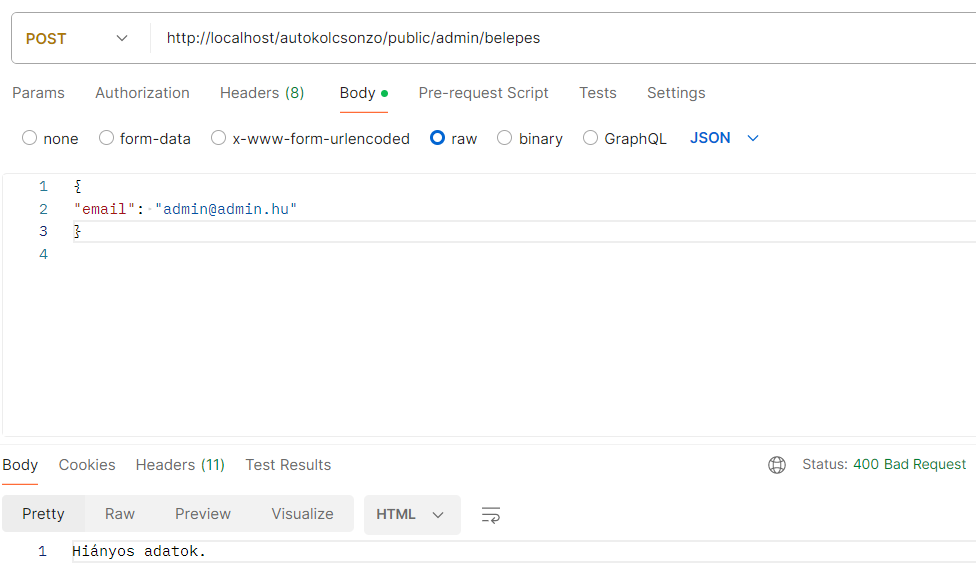
##### 2.2.3.1. A Belépés tesztelése

Jelen pontban a felhasználó Belépés funkció kerül megvizsgálásra. A művelet végrehajtásához a <http://localhost/autokolcsonzo/public/admin/belepes> végpontot kell használnunk. A beérkező metódus POST. A belépéskor 2 adatot kell megadni az e-mail címet és a jelszót. A tesztelés során az e-mail: [admin@admin.hu](mailto:admin@admin.hu) és a jelszó: admin párost fogjuk hasznáni. Ha megfelelő a e-mail.cim és jelszó páros akkor a felhasználó id-jével (72) tér vissza és 200-as státusz kóddal és OK üzenettel. Vizsgáljuk meg lépésről lépésre.

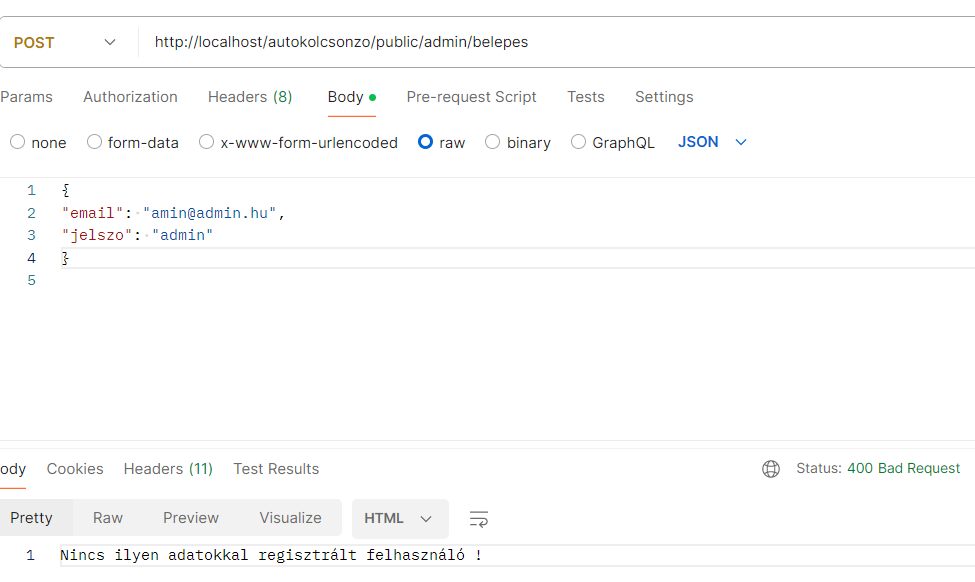
Teszt 1: A függvény először megvizsgálja, hogy milyen metódussal érkezett a kérés. Állítusk be pl a GET metódust.

10. ábra: Belépés tesztelése POSTMAN-nel 1.

Ahogy vártuk a függvény nem fut le, hanem megáll, és ***Nem megfelelő Methódus típus*** üzenettel és ***400 Bad Request*** státusszal tér vissza.

Teszt 2: Amikor POST metódussal érkezik a kérés akkor a következő, hogy csak akkor engedi folytatni a belépést, ha mindkettő azonosítót, e-mailt és a jelszót is megadja a felhasználó. Ebben az esetben ***Hiányos üzenettel*** és ***400 Bad Request*** státusszal kell visszatérnie.

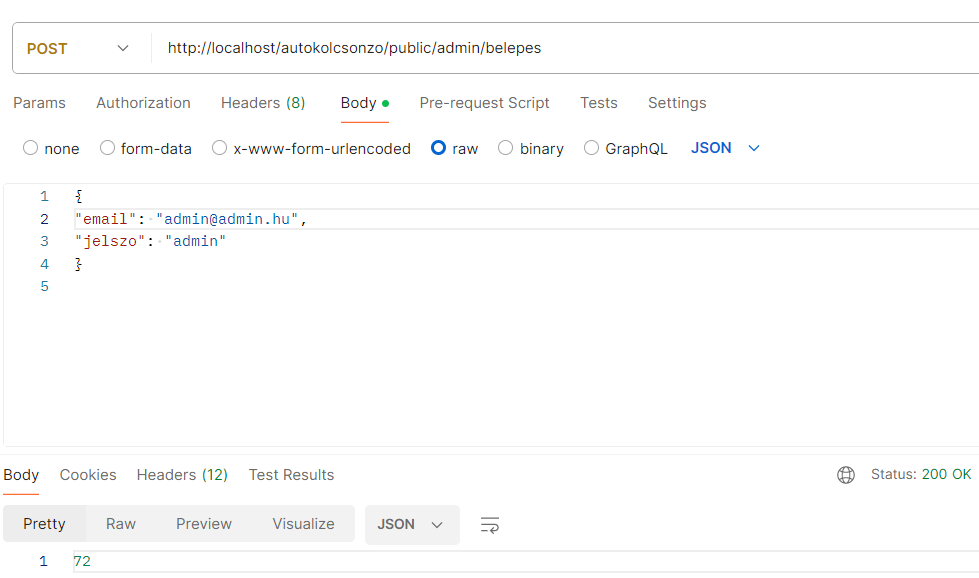
11. ábra: Belépés tesztelése POSTMAN-nel 2.

Teszt 3: Ebben az esetben POST metódussal érkezik a kérés és a felhasználó megadta a két szükséges adatot. Most azt vizsgáljuk, hogy a felhasználó megtalálható az adatbázisban ezzel az e-mail címmel. Ha nincs ilyen regisztrált e-mail címmel felhasználó az adatbázisban, akkor ***Nincs ilyen adatokkal regisztrált felhasználó!*** és ***400 Bad Request*** státusszal kell visszatérnie.

12. ábra: Belépés tesztelése POSTMAN-nel 3.

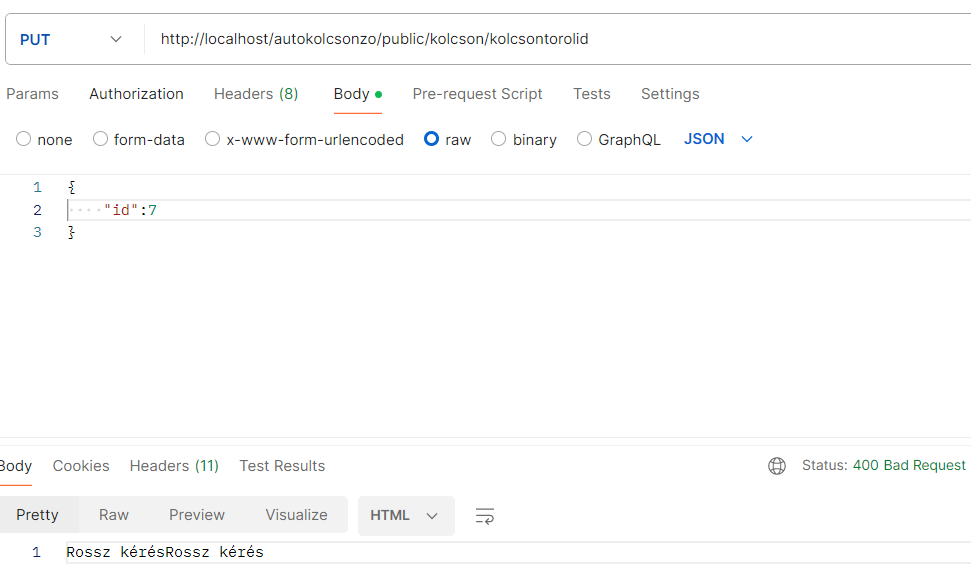
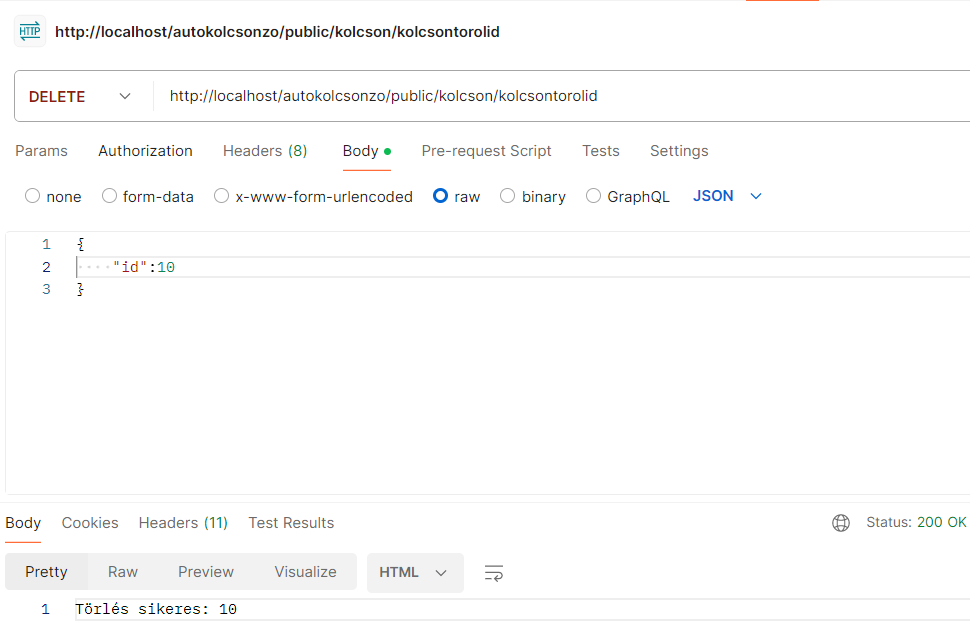
Teszt 4: Végül nézzük meg azt az esetet, amikor a kérés POST metódussal érkezik, a felhasználó regisztálva van az adatbázisban az e-mail címmel, és jó a jelszava is.

13. ábra: Belépés tesztelése POSTMAN-nel 4 .

Ahogy látható a kérés eredménye, amit vártunk is a felhasználó ID-ja (72) illetve a Status 200 OK.

##### 2.2.3.2. A Foglalás törlésének tesztelése

Teszt 1: Most az autó foglalás törlésének a funkciójat vizsgáljuk. A művelet végrehajtásához a <http://localhost/autokolcsonzo/public/kolcson/kolcsontorolid> végpontot használjuk. A kérésből kapjuk meg a törlendő foglalás id-jét és a metódusát. Miután adat törlésről van szó így a metódus DELETE. Mivel törlés akkor is végrehajtódik, ha nincs olyan id-jű foglalás így csak a metódust vizsgáljuk meg. Ha nem DELETE metódussal érkezik a kérés akkor ***400 Bed Request*** státusszal és ***Rossz kérés*** üzenettet kapunk. Ha a DELETE metódussal érkezik a kérés akkor a modell a *delkolcsonzes( $id )* függvényt hívja meg és a foglalást törli a kolcsonzes táblából. Az üzenet, ami visszaérkezik ***200 OK*** státusszal és a törölt foglalás ID.jével ***Törlés sikeres: .$id.***



15. ábra: Foglalás törlésének tesztelése POSTMAN-nel helyes metódussal

14. ábra: Foglalás törlésének tesztelése POSTMAN-nel rossz metódussal

2.3.2. FRONTEND

2.3.2.1 Navbar Komponens leírása

A navbar egy grafikus felhasználói felület (GUI) eleme, amelyet a felhasználók a weboldal különböző részei közötti navigáláshoz használnak. A weboldal tetején helyezkedik el, és vízszintes. A navbaron linkek, gombok és más elemek találhatók, amelyek a felhasználót a weboldal különböző funkcióihoz irányítják.

***1. Adatok:***

Navlinks, NavlinksLogin, Navlinks2: Ezek konstans tömbök, amelyek tartalmazzák a navigációs linkek adatait, mint a név, az id és a hivatkozás. A listaid sessionStorage-ból lekéri a felhasználói azonosítót.

A useState hook használata az adatok, nev, és showMenu állapotok létrehozásához. Az adatok állapot tárolja a bejelentkezett felhasználó adatait, a nev az aktuális felhasználó nevét, a showMenu pedig a mobil menü megjelenítési állapotát.

***2. Fetch kérés:***

A kód ellenőrzi a listaid értékét. Ha van érték, akkor egy fetch kérést hajt végre a szerverhez a felhasználói adatok lekérése érdekében. A válasz JSON formátumban érkezik, melyet az adatok állapotban tárol.

***3. Navigáció:***

A useNavigate hook segítségével történik az oldalak közötti navigáció.

A Link komponens (valószínűleg a Next.js routerből) használatos a linkek kezelésére.

A kilepes függvény a kijelentkezést valósítja meg, törli a listaid értéket a sessionStorage-ból és a navigate segítségével az oldal átirányítása a főoldalra történik.

***4. Menü megjelenítés:***

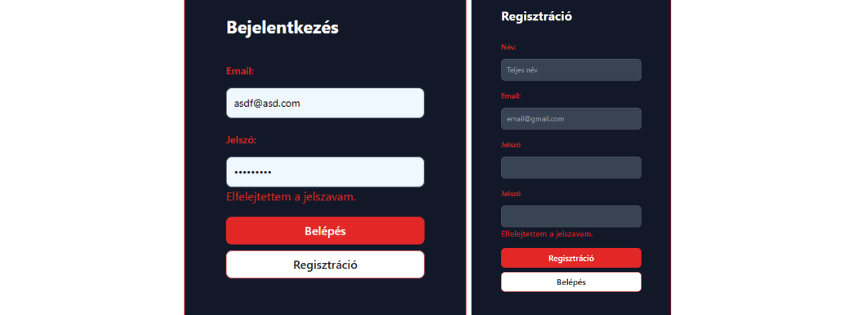
A showMenu állapot határozza meg a mobil menü megjelenését.

A hamburger menü ikonok a HiMenuAlt1 és HiMenuAlt3 komponensekből (valószínűleg egy ikon könyvtárból) kerülnek megjelenítésre az állapot értékétől függően.

***5. ResponsiveMenu komponens:***

A kód egy ResponsiveMenu komponenensre hivatkozik, amely valószínűleg a mobil menü logikáját és megjelenítését tartalmazza.

Összességében ez a kód egy dinamikus navbar komponenst valósít meg, amely a felhasználó bejelentkezési állapota alapján változtatja a megjelenített linkeket, és mobil nézetben egy hamburger menüt használ a navigációhoz.

2.3.2.2 Regisztráció, Belépés komponensek leírása:

16. ábra: Bejelentkezés és a Regisztráció komponensek

Regisztráció Komponens:

A Register komponens az alábbi állapotváltozókat használja:

* nev: A felhasználó neve.
* jelszo: A felhasználó jelszava.
* jelszo2: A jelszó megerősítése.
* email: A felhasználó e-mail címe.

A komponens tartalmaz egy űrlapot, amely a következő mezőket tartalmazza:

* Név
* E-mail cím
* Jelszó
* Jelszó megerősítése

A felhasználó beírja az adatait az űrlap mezőibe, majd a "Regisztráció" gombra kattint a benyújtáshoz.

Az onSubmit függvény kezeli az űrlap benyújtását. Ellenőrzi, hogy a jelszó és a jelszó megerősítése megegyezik-e. Ha igen, akkor a kuldes függvényt hívja meg, amely elküldi a felhasználó adatait a szervernek a regisztráció befejezéséhez.

A kuldes függvény a fetch API-t használja a felhasználó adatainak POST kéréssel történő elküldéséhez a szervernek. A válasz alapján a komponens megjeleníti a sikeres regisztrációt, vagy hibaüzenetet jelenít meg, ha a regisztráció sikertelen.

Sikeres regisztráció esetén a komponens átirányítja a felhasználót a bejelentkezési oldalra.

A komponens a react-router-dom csomag Link komponensét használja az oldalak közötti navigációhoz.

A komponens az useState hookot használja az állapotváltozók kezelésére.

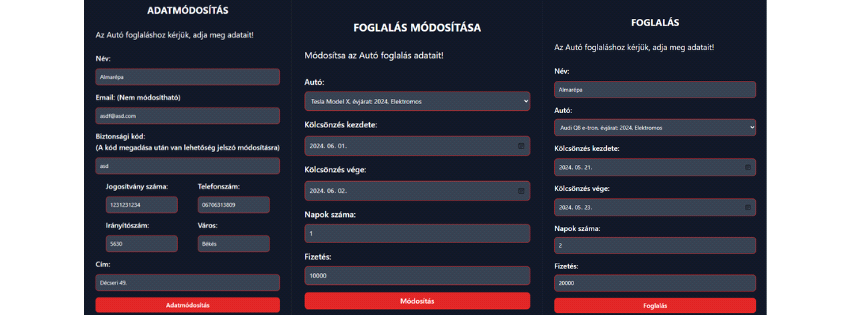
A komponens az useNavigate hookot használja az oldalak közötti navigációhoz.

Belépés Komponens:

formData: Objektum, amely az email és jelszó mezők értékeit tárolja. (email: string, jelszo: string)

Függvények:

* kuldes(formData, method): Elküld egy fetch kérést a megadott formData objektummal és method HTTP módszerrel a <http://localhost/autokolcsonzo/public/admin/belepes> végpontra. A válasz alapján beállítja a sessionStorage-t, megjelenít egy értesítést és navigál az alkalmazás kezdőlapjára, vagy hiba esetén toast értesítést jelenít meg.
* onSubmit(e): Eseménykezelő az űrlap elküldéséhez. Megakadályozza az alapértelmezett form submit viselkedést, meghívja a kuldes függvényt a POST metódussal és a formData állapottal, majd navigál az alkalmazás kezdőlapjára.
* writeData(e): Eseménykezelő az űrlapmezők változásához. Frissíti a formData állapotot az aktuális érték alapján.
* JSX: A komponens egy űrlapot renderel, amely két mezőt tartalmaz az email cím és a jelszó megadásához. Bejelentkezés gomb és regisztrációs link található az űrlap alatt.

2.3.2.3 Adatmódosítás, Foglalás, Foglalás módosítás komponensek leírása:

17. ábra: Adatmódosítás, Foglalás módosítása, Foglalás komponensek

Adatmódosítás Komponens:

**Importált modulok:**

useState, useEffect: A React állapotkezelő hook-jai.

useNavigate: A react-router-dom navigációs eszköze.

UserContext: A felhasználói adatokat és függvényeket tartalmazza a context provider-ből.

Komponens logikája:

**Inicializálás:**

A sessionStorage.getItem('id') segítségével lekéri a felhasználó azonosítóját.

A useContext(UserContext) segítségével eléri a refresh és logout függvényeket a UserContext-ből.

Az useNavigate hook segítségével később navigálhat a komponens.

Létrehoz két állapotváltozót az useState hook segítségével:

* adatok: Üres tömb, ami majd a felhasználó adatait fogja tartalmazni.
* formData: Objektum, amely a kitöltött űrlap adatait tárolja.

**Adatok lekérése:**

Az useEffect hook segítségével lekéri a felhasználó adatait az adatbázisból.

A fetch API használatával küld egy POST kérést a <http://localhost/autokolcsonzo/public/admin/listaidmod> végpontra.

A kérés body-jában elküldi a listaidmod változó értékét, ami a felhasználó azonosítója.

A válasz JSON formátumban érkezik.

A válaszban kapott adatokat beállítja az adatok állapotváltozóba.

Az első elemet (adatok[0]) beállítja a formData állapotváltozóba, így az űrlap eleinte a felhasználó aktuális adataival jelenik meg.

Hiba esetén megjelenít egy alert üzenetet.

**Űrlap kezelése:**

A writeFormData függvény frissíti a formData állapotváltozót az űrlap mezőiben történt módosítások alapján.

Az onChange eseménykezelő figyeli az űrlap mezők változását, és az új értéket az id attribútum alapján a megfelelő mezőbe menti a formData objektumban.

**Adatok módosítása:**

Az adatkuldes függvény a kitöltött űrlap adatait elküldi a szervernek módosításra.

fetch API használatával küld egy POST kérést a <http://localhost/autokolcsonzo/public/admin/adatmodosit> végpontra.

A kérés body-jában elküldi a formData objektumot JSON formátumban.

Sikeres módosítás esetén megjelenít egy alert üzenetet.

Hiba esetén console-ba logolja a hibát, és megjelenít egy általános alert üzenetet a felhasználónak.

**Űrlap beküldése:**

Az onSubmit függvény az űrlap beküldésekor fut le.

Megakadályozza az alapértelmezett űrlapküldést az e.preventDefault() metódus segítségével.

Meghívja az adatkuldes függvényt az adatok elküldésére.

Üres értékekre állítja a formData állapotváltozót.

A navigate('/carlist') segítségével navigál a "/carlist" oldalra.

**JSX kód:**

A Modositas komponens renderel egy div elemet, amely tartalmazza az űrlapot.

Az űrlap mezői a felhasználó nevét, jogosítvány számát, telefonszámát, születési idejét, biztonsági kódját, email címét (letiltva - nem módosítható), irányítószámát, városát és címét kérik le.

Foglalás Komponens:

**A listaidmod:**

Ez a változó a sessionStorage-ből lekérdezett azonosítót tárolja, és a felhasználó adatainak lekérésére szolgál.

* adatok: Üres tömb a felhasználói adatok tárolására.
* options: Üres tömb az autóadatok tárolására a legördülő menü számára.
* loading: Inicializáláskor true értéket kapó állapotváltozó, ami jelzi az adatok betöltését.
* formData: A felhasználói információkat tartalmazó űrlapmezőket tartalmazó objektum.
* formrendeles: Az autókölcsönzési adatokat tartalmazó űrlapmezőket tartalmazó objektum.

**Adatok lekérése:**

Két useEffect hook-ot használunk az adatok leküldésére:

Az első a listaidmod azonosító segítségével a felhasználó adatait kérdezi le. Feltölti az adatok tömböt, és az adatok első eleme alapján állítja be a formData objektum kezdeti értékeit.

A második lekéri az autóadatokat, és feltölti az options tömböt az autó kiválasztási legördülő menü számára.

Feltételes megjelenítés:

A komponens feltételesen renderel tartalmat a loading állapot alapján. Adatbeolvasás közben a "Betöltés..." üzenetet jeleníti meg.

**Űrlap kezelés:**

A writeFormRendeles függvény a felhasználói beviteli változtatások alapján frissíti a formrendeles állapotobjektumot.

Az onSubmit függvény kezeli az űrlap beküldését:

A adatkuldes függvényt hívja meg a foglalási adatok elküldéséhez.

Sikeres beküldés esetén megjelenít egy alert üzenetet, és a navigate segítségével a "rendelesek" oldalra navigál.

Hiba esetén egy alert segítségével jelenít meg hibaüzenetet.

**Űrlapmezők:**

Az űrlap különféle felhasználói és autókölcsönzési adatokat tartalmazó mezőket tartalmaz:

Felhasználói adatok:

* Név (nev)
* Jogosítvány szám (jogositvanyzama)
* Telefonszám (telefonszam)
* Születési idő (szuletesiido)
* Biztonsági kód (biztonsagikod)
* E-mail cím (email)
* Cím (isz, varos, cim)
* Autókölcsönzési adatok:
* Ügyfél azonosító (berloid) (előre kitöltve a listaidmod-ból)
* Autó azonosító (autoid) (legördülő menüből választva)
* Kölcsönzés kezdete (berletkezdete)
* Kölcsönzés vége (berletvege)
* Napok száma (napokszama) (automatikusan kiszámítva a kezdő és befejező dátum alapján)
* Napi díj (napidij) (automatikusan kiszámítva a napok száma alapján)

Hookokat használ az adatlekéréshez, állapotkezeléshez és feltételes rendereléshez.

Foglalás módosítás Komponens:

**Importált modulok:**

useState, useEffect: A React állapotkezelő hook-jai.

useNavigate: A react-router-dom navigációs eszköze.

useContext: A UserContext-ből való adat eléréséhez.

**Inicializálás:**

A sessionStorage.getItem('id') segítségével lekéri a felhasználó azonosítóját.

A useContext(UserContext) segítségével eléri a refresh és logout függvényeket a UserContext-ből.

A useNavigate hook segítségével később navigálhat a komponens.

Létrehoz két állapotváltozót a useState hook segítségével:

* adat: Üres objektum, ami majd a lefoglalt autó eredeti adatait fogja tartalmazni.
* formData: Objektum, amely a módosítani kívánt adatokat tárolja.

**Adatok lekérése:**

Az useEffect hook segítségével lekéri a felhasználó által lefoglalt autó adatait az adatbázisból.

A fetch API használatával küld egy POST kérést a <http://localhost/autokolcsonzo/public/autober/lista> végpontra.

A válasz JSON formátumban érkezik.

A választ tartalmazó objektumot beállítja az adat állapotváltozóba.

Az adat objektum alapján beállítja a formData állapotváltozót az eredeti értékekre.

Hiba esetén megjelenít egy alert üzenetet.

**Adatmódosítás:**

A writeFormData függvény frissíti a formData állapotváltozót az űrlap mezőiben történt módosítások alapján.

Az onChange eseménykezelő figyeli az űrlap mezők változását, és az új értéket az id attribútum alapján a megfelelő mezőbe menti a formData objektumban.

**Űrlap beküldése:**

Az adatkuldes függvény a módosított adatokat elküldi a szervernek frissítésre.

A fetch API használatával küld egy POST kérést a <http://localhost/autokolcsonzo/public/kolcson/kolcsonmod> végpontra.

A kérés body-jában elküldi a formData objektumot JSON formátumban.

Sikeres módosítás esetén megjelenít egy alert üzenetet.

Hiba esetén console-ba logolja a hibát, és megjelenít egy általános alert üzenetet a felhasználónak.

**Az űrlap beküldése után:**

Üres értékekre állítja a formData állapotváltozót.

A navigate('/rendelesek') segítségével navigál a "/rendelesek" oldalra.

**Megjelenítés:**

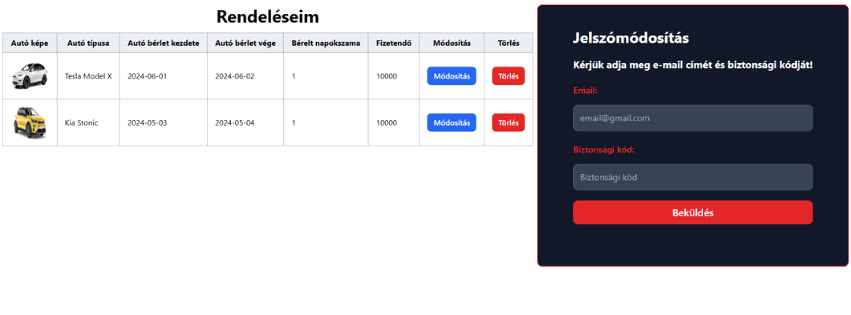
A komponens renderel egy div elemet, amely tartalmazza a foglalás adatainak módosítására szolgáló űrlapot.

Az űrlap mezői az alábbi adatok módosítását teszik lehetővé:

Autó

* Kölcsönzés kezdete (dátum)
* Kölcsönzés vége (dátum)
* Napok száma (automatikusan kitöltődik a kezdő és befejező dátum alapján)
* Fizetendő összeg (automatikusan kitöltődik a kölcsönzés időtartama alapján)

Az űrlap beküldése után a rendszer frissíti a foglalás adatait, és a felhasználót a rendeléseket listázó oldalra navigálja.

2.3.2.4 Emlékeztető és Rendelések komponensek leírása: 

18. ábra: Rendeléseim és az Emlékeztető komponensek

Emlékeztető Komponens:

**Importált modulok:**

useState: A React állapotkezelő hook-ja.

useNavigate: A react-router-dom navigációs eszköze.

**Inicializálás:**

Az üres state változók deklarálása:

* nev: Üres tömb a név tárolására (jelenleg nincs használatban).
* jelszo: Üres string a jelszó tárolására.
* jelszo2: Üres string a jelszó megerősítésére.
* email: Üres string az email cím tárolására.
* biztonsagikod: Üres string a biztonsági kód tárolására.
* id: Szám típusú változó, alapértelmezetten 0, a felhasználó azonosítójának tárolására szolgál.

Az useNavigate hook segítségével később navigálhat a komponens.

Első lépés: E-mail cím és biztonsági kód megadása:

A komponens rendereléskor ellenőrzi az id state értékét.

Ha az id értéke 0, akkor az első lépést jeleníti meg:

A felhasználónak meg kell adnia az email címét és a biztonsági kódot.

Beküldés gombbal ellátott az űrlap. Az űrlap beküldésekor az onSubmit függvény fut le.

Az onSubmit függvény ellenőrzi, hogy van-e már érték az id state változóban.

Ha nincs érték (id értéke 0), akkor a kuldes függvényt hívja meg.

Ha van érték (id értéke nagyobb, mint 0), akkor ellenőrzi a két jelszó megegyezését.

Ha a jelszavak megegyeznek, akkor a kuldesujjelszo függvényt hívja meg.

Ha a jelszavak nem egyeznek, akkor alert üzenet jelenik meg a hibáról.

Első lépés: Adatok küldése a szervernek (biztonsági kód ellenőrzés):

A kuldes függvény a fetch API használatával POST kérést küld a szervernek a megadott email cím és biztonsági kód alapján történő azonosításhoz.

A válasz JSON formátumban érkezik.

Sikeres azonosítás esetén a válasz tartalmazza a felhasználó azonosítóját (id).

**Sikeres válasz esetén:**

Az id state értéke beállítódik a kapott értékre.

A komponens a második lépést jeleníti meg.

Sikertelen azonosítás esetén:

Alert üzenet jelenik meg a hibáról.

Második lépés: Új jelszó megadása:

Ha az id state értéke nagyobb, mint 0, akkor a második lépést jeleníti meg:

A felhasználónak meg kell adnia az új jelszót kétszer.

Beküldés gombbal beküldhető az űrlap.

Az űrlap beküldésekor ismét az onSubmit függvény fut le.

Mivel az id state értéke már be van állítva, ezért a jelszó módosítására kerül sor.

Második lépés: Jelszó módosítása:

A kuldesujjelszo függvény async/await szintaxis használatával küldi a módosítási kérést a szervernek.

A kérés a felhasználó azonosítóját (id) és az új jelszót (jelszo) tartalmazza.

**Sikeres módosítás esetén:**

Alert üzenet jelenik meg a sikeres jelszómódosításról.

Rendelések Komponens:

**Importált modulok:**

useState: A React állapotkezelő hook-ja.

useEffect: A React mellékhatás-kezelő hook-ja.

useNavigate: A react-router-dom navigációs eszköze.

UserContext: Kontextus a felhasználói adatok eléréséhez.

useContext: A React hook a kontextus eléréséhez.

**Inicializálás:**

A state változók deklarálása:

berloid: A sessionStorage-ből kiolvassa a felhasználó azonosítóját.

adatok: Üres tömb a kölcsönzési adatok tárolására.

A useContext hook segítségével eléri az UserContext értékeit:

refresh: A felhasználói adatok frissítésére szolgáló függvény (jelenleg nincs használatban).

A useNavigate hook segítségével később navigálhat a komponens.

**Adatok lekérése:**

Az useEffect hook segítségével az első rendereléskor lefut a benne foglalt függvény.

A fetch API használatával POST kérést küld a szervernek a /autokolcsonzo/public/kolcson/rendelesek végpontra.

A kérés tartalmazza a felhasználó azonosítóját (berloid).

Sikeres válasz esetén a JSON formátumú válasz tartalmazza a felhasználó összes kölcsönzési adatát. A kapott adatokat az adatok state változóba menti.

Sikertelen kérés esetén alert üzenet jelenik meg a hibáról.

**Rendelések megjelenítése:**

A komponens rendereléskor a adatok state értékétől függően jeleníti meg a kölcsönzéseket.

Ha az adatok tömb nem üres, akkor egy táblázatot jelenít meg a kölcsönzési adatokkal:

A táblázat fejléce tartalmazza az autók adatait (kép, típus, bérleti időszak, bérleti díj) és a módosítás/törlés lehetőségeket.

A táblázat törzse a felhasználó összes kölcsönzését soronként jeleníti meg.

Minden sor tartalmazza az autó képét, típusát, bérleti időszakát, bérleti díját, valamint két gombot:

Módosítás gomb: A gombra kattintva a felhasználó a foglalás módosító oldalára navigál az adott kölcsönzés adataival.

Törlés gomb: A gombra kattintva a torles függvény fut le.

**Törlés funkció:**

A torles függvény a fetch API használatával DELETE kérést küld a szervernek a /autokolcsonzo/public/kolcson/kolcsontorolid végpontra.

A kérés tartalmazza a törölni kívánt kölcsönzés azonosítóját (id).

Sikeres válasz esetén megjelenik egy alert üzenet a sikeres törlésről, és a komponens frissül a legfrissebb adatokkal.

Sikertelen kérés esetén alert üzenet jelenik meg a hibáról.

2.3.2.5 Ügyféladatlap és ResponsivMenü komponensek leírása:

Ügyféladatlap komponens:

**Importált modulok:**

useState: React állapotkezelő hook

useEffect: React mellékhatás-kezelő hook

useNavigate: React Router-dom navigációs eszköze

useContext: React kontextus elérésének hook-ja

UserContext: Kontextus a felhasználói adatok eléréséhez (opcionális)

Link: React Router-dom link komponens

**Inicializálás:**

A useState hook segítségével az alábbi állapotváltozókat hozzuk létre:

* listaidmod: A sessionStorage-ből kiolvassa az ügyfél azonosítóját.
* adatok: Üres tömb az ügyfél adatainak tárolására.
* formData: Objektum az űrlapmezők adatainak tárolására (név, jogosítvány száma, telefonszám stb.).

Ha az UserContext létezik, akkor a useContext hook segítségével lekérdezzük a refresh és logout függvényeket a felhasználói adatok frissítéséhez és a kijelentkezéshez (opcionális).

A useNavigate hook segítségével később navigálhatunk a komponensből.

**Adatok lekérése:**

Az useEffect hook segítségével az első rendereléskor lefut a benne foglalt függvény.

A fetch API használatával POST kérést küldünk a szervernek a /autokolcsonzo/public/admin/listaidmod végpontra.

A kérés tartalmazza az listaidmod állapotváltozó értékét, vagyis az ügyfél azonosítóját.

A kérés fejlécében megadjuk, hogy JSON formátumú adatot várunk vissza (Content-type: application/json).

Sikeres válasz esetén a JSON formátumú válasz az ügyfél összes adatát tartalmazza.

A kapott adatokat az adatok állapotváltozóba mentjük.

Az űrlapmezők kitöltése érdekében az első talált elemet az formData állapotváltozóba másoljuk (setFormData(adatok[0])).

Sikertelen kérés esetén alert üzenet jelenik meg a hibáról.

**Adatok megjelenítése:**

A komponens renderelésekor a adatok állapotváltozó értékétől függően jelenítjük meg az ügyfél adatait.

Ha az adatok tömb nem üres, akkor egy űrlapot jelenítünk meg az ügyfél összes adatával:

Az űrlapmezők a következőket tartalmazzák: név, jogosítvány száma, telefonszám, születési idő, biztonsági kód, email cím, irányítószám, város, cím.

A legtöbb mező kitöltött, de a biztonsági kód mezőt csak az űrlap elküldése előtt szabad kitölteni.

Az űrlap jelenleg nem módosítja az adatbázist (hiányzik a frissítési logika).

Ha az adatok tömb üres, akkor semmit sem jelenítünk meg, vagy egy üzenettel jelezhetjük, hogy nincsenek adatok.

ResponsivMenu komponens:

Ez a React komponens egy hamburger menüt valósít meg, amely telefonos nézetben jelenik meg, és az oldal főbb navigációs hivatkozásait tartalmazza. A bejelentkezett felhasználók számára a név és a kilépési lehetőség is megjelenik.

**Importált modulok:**

useState: React állapotkezelő hook

useEffect: React mellékhatás-kezelő hook

useNavigate: React Router-dom navigációs eszköze

useContext: React kontextus elérésének hook-ja

UserContext: Kontextus a felhasználói adatok eléréséhez (opcionális)

Link: React Router-dom link komponens

FaUserCircle: React Icons ikon (felhasználó ikon)

Navlinks: Navigációs linkek tömbje (be nem jelentkezett felhasználóknak)

Navlinks2: Navigációs linkek tömbje (bejelentkezett felhasználóknak)

**Inicializálás:**

A useState hook segítségével az alábbi állapotváltozókat hozzuk létre:

* listaid: A sessionStorage-ből kiolvassa az ügyfél azonosítóját.
* adatok: Üres tömb az ügyfél adatainak tárolására.
* nev: Az aktuálisan bejelentkezett felhasználó neve (üres kezdetben).

Ha az UserContext létezik, akkor a useContext hook segítségével lekérdezzük a refresh és logout függvényeket a felhasználói adatok frissítéséhez és a kijelentkezéshez (opcionális).

A useNavigate hook segítségével később navigálhatunk a komponensből.

**Adatok lekérése:**

Az useEffect hook segítségével az első rendereléskor lefut a benne foglalt függvény, feltéve hogy van listaid (azaz be van jelentkezve a felhasználó).

A fetch API használatával POST kérést küldünk a szervernek a /autokolcsonzo/public/admin/listaid végpontra.

A kérés tartalmazza az listaid állapotváltozó értékét, vagyis az ügyfél azonosítóját.

A kérés fejlécében megadjuk, hogy JSON formátumú adatot várunk vissza (Content-type: application/json).

Sikeres válasz esetén a JSON formátumú válasz az ügyfél összes adatát tartalmazza.

A kapott adatokból kiolvassuk a nevet, és az nev állapotváltozóba mentjük.

Az összes adatot az adatok állapotváltozóba is mentjük (bár jelenleg nem használjuk a komponens rendereléséhez).

Sikertelen kérés esetén alert üzenet jelenik meg a hibáról.

**Menü megjelenítése:**

A komponens a showMenu prop alapján jelenik meg vagy tűnik el (külső komponensből vezérelhető).

A megjelenített tartalom attól függ, hogy be van-e jelentkezve a felhasználó:

Be nem jelentkezett felhasználó esetén a Navlinks tömb elemeit jeleníti meg, amelyek általános navigációs linkeket tartalmaznak.

Bejelentkezett felhasználó esetén a Navlinks2 tömb elemeit jeleníti meg, amelyek szintén navigációs linkeket tartalmaznak, kiegészítve a felhasználó nevével és a kilépési lehetőséggel (Kilépés).

**Funkciók:**

alma funkció: Az ügyfél adatok oldalára navigál (/ugyfeladatok), majd frissíti az oldalt a window.location.reload() segítségével.

kilepes funkció: Törli a listaid értéket a sessionStorage-ből, a logout függvényt meghívja ( a UserContext kontextuson keresztül), majd a főoldalra navigál (/) és frissíti az oldalt.

2.3.2.6 CarList, Összesek és Összes komponensek:  
CarList Komponens:

Ez a React komponens egy autólista megjelenítését és szűrését teszi lehetővé. A felhasználó választhatja, hogy az összes márkát szeretné látni, vagy csak néhány konkrét típust (Ford, Audi, Tesla, Mercedes, Kia).

**Importált modulok:**

useState React állapotkezelő hook

Osszesek: Komponens az összes autó megjelenítéséhez ("./Osszesek")

Kiak: Komponens a Kia autók megjelenítéséhez ("./Kiak")

Mercik: Komponens a Mercedes autók megjelenítéséhez ("./Mercik")

Teslak: Komponens a Tesla autók megjelenítéséhez ("./Teslak")

Audik: Komponens az Audi autók megjelenítéséhez ("./Audik")

Fordok: Komponens a Ford autók megjelenítéséhez ("./Fordok")

**Komponens logikája:**

Állapotváltozók:

* isShowMore: Boolean állapotváltozó, amely meghatározza, hogy a Fordok komponens megjelenjen-e (alapértelmezésben false).
* isShowMores: Boolean állapotváltozó, amely meghatározza, hogy az Audik komponens megjelenjen-e (alapértelmezésben false).
* isShowMoress: Boolean állapotváltozó, amely meghatározza, hogy a Teslak komponens megjelenjen-e (alapértelmezésben false).
* isShowMoresss: Boolean állapotváltozó, amely meghatározza, hogy a Mercik komponens megjelenjen-e (alapértelmezésben false).
* isShowMoressss: Boolean állapotváltozó, amely meghatározza, hogy a Kiak komponens megjelenjen-e (alapértelmezésben false).
* isShowMoresssss: Boolean állapotváltozó, amely meghatározza, hogy az Osszesek komponens megjelenjen-e (alapértelmezésben true).

**Szűrés gombok:**

A komponens hat gombot tartalmaz, amelyek mindegyike egy márkához tartozik (Ford, Audi, Tesla, Mercedes, Kia, Összes).

Minden gombhoz tartozik egy toggleReadMore... függvény, amely az adott márka isShowMore... állapotváltozóját állítja át az ellenkezőjére.

Az Összes gomb kivétel, az alapértelmezés szerint mindig be van kapcsolva. A többi gomb megnyomása esetén az Összes gomb kikapcsolódik.

**Komponensek megjelenítése:**

A komponens megjeleníti a kiválasztott márkák komponenseit a useState hook segítségével.

Minden márkához tartozik egy feltételes megjelenítés (isShowMore... &&):

Ha az isShowMore... állapotváltozó true, akkor az adott márka komponense megjelenik.

Ha false, akkor a komponens rejtve marad.

2.3.2.7. Összesek Komponens:  
Ez a React komponens az összes elérhető autó adatait jeleníti meg. Az adatok egy API végpont (<http://localhost/autokolcsonzo/public/osszes/lista>) segítségével kerülnek lekérésre.

**Importált modulok:**

useState: React állapotkezelő hook

useEffect: React mellékhatás-kezelő hook

Osszes: Komponens az egyes autók részletes adatainak megjelenítéséhez ("./Osszes")

**Komponens logikája:**

Állapotváltozó:

Autok: Inicializáláskor üres tömb ([]), később az API-ból lekért autók adatait tartalmazza.

Adatok lekérése:

Az useEffect hook segítségével az első rendereléskor lefut a benne foglalt függvény.

A fetch API használatával GET kérést küldünk a <http://localhost/autokolcsonzo/public/osszes/lista> végpontra.

Sikeres válasz esetén a válasz JSON formátumú tömböt tartalmaz, amely az összes autó adatait tartalmazza.

A lekért adatokat az Autok állapotváltozóba mentjük.

Sikertelen kérés esetén hibaüzenet jelenik meg alert formában.

Autók megjelenítése:

A komponens a map metódus segítségével bejárja az Autok tömböt.

Minden egyes autóra (az Auto konstansban) lefuttatja az Osszes komponenst, amelynek a key prop-ját az aktuális indexre állítja. Így az egyes autók komponensei egyedi kulccsal rendelkeznek.  
2.3.2.8. Összes Komponens:

Ez a React komponens egyetlen autó adatait és részletes tulajdonságait jeleníti meg. A komponens bemenete az Auto objektum, amely az autó adatait tartalmazza.

**Importált modulok:**

useState: React állapotkezelő hook

Komponens logikája:

**Állapotváltozó:**

isShowMore: Boolean állapotváltozó, amely meghatározza, hogy a teljes adatlista megjelenjen-e (alapértelmezésben false).

**Adatok megjelenítése:**

A komponens megjeleníti az Auto objektum tulajdonságait egy strukturált formában.

A komponens fejléceként az Auto.tipus érték szerepel.

Az autó képe az Auto.kepek URL alapján kerül megjelenítésre.

A "Több" gombra kattintva további részletek válnak láthatóvá az autóról. Ezek az adatok az Auto objektum következő tulajdonságaiból származnak:

* biztonság: Az autó biztonsági jellemzői
* ülések: Az ülések száma
* klima: A klímaberendezés típusa
* audio: Az audiorendszer típusa
* belsofelszereltseg: A belső felszereltség felsorolása
* valto: A sebességváltó típusa
* vezetesttamogatorendszerek: A vezetés támogató rendszerek felsorolása
* motor: A motor típusa
* uzemanyag: Az autó üzemanyagtípusa
* "Több" gomb:

A komponens tartalmaz egy "Több" gombot, amely az isShowMore állapotváltozót kapcsolja.

A gomb felirata a isShowMore értékétől függően változik:

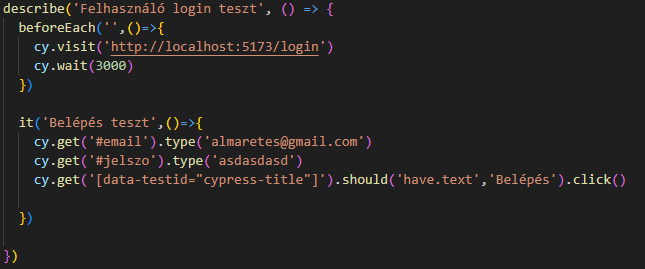
Ha false, akkor a gomb felirata "Több" (alapértelmezett). A gomb megnyomásakor a további részletek láthatóvá válnak.

Ha true, akkor a gomb felirata "Kevesebb". A gomb megnyomásakor a további részletek rejtve maradnak.

**Megjegyzés:**

Összes többi autó komponens lekérdezése és megjelenítése is ezen az elven működik.

2.4. Tesztdokumentáció

2.4.2.1.A sikeres bejelentkezés vizsgálata.

19. ábra: Felhasználó login teszt

**Ez a teszt leírás az alábbiakat vizsgálja:**

Meg tud-e jelentkezni a felhasználó a <http://localhost:5173/login> címen elérhető felületen érvényes e-mail címmel és jelszóval.

**Előkészületek:**

A teszt minden futtatása előtt meglátogatja a <http://localhost:5173/login> címet.

Vár 3 másodpercet, hogy a bejelentkezési felület betöltődjön.

***Teszt esetek:***

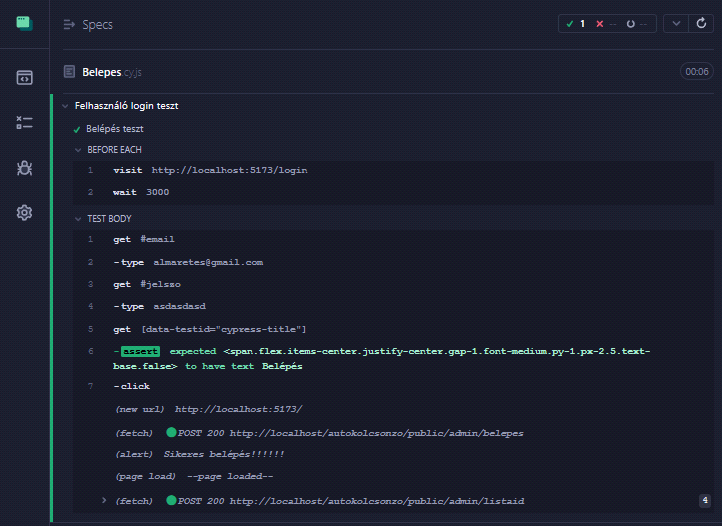
**1. Belépés teszt:**

Beírjuk az almaretes@gmail.com e-mail címet az #email beviteli mezőbe.

Beírjuk az asdasdasd jelszót az #jelszo beviteli mezőbe.

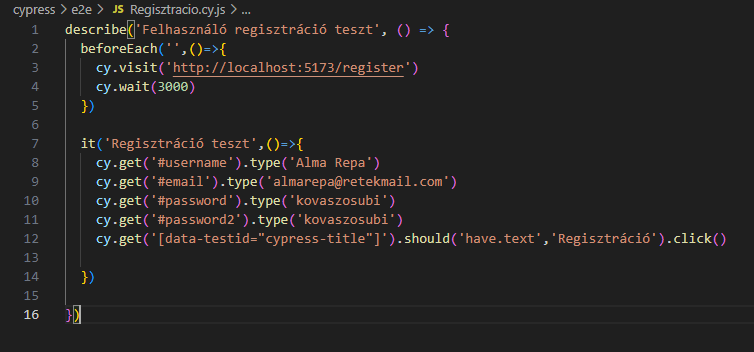
Kattintunk a "Belépés" gombra, amelynek data-testid attribútuma cypress-title.

Elvárás: A teszt sikeresnek minősül, ha a felhasználó sikeresen bejelentkezik, és a felület címe megváltozik.



20. ábra: Felhasználó login teszt eredménye

2.4.2.2.A sikeres regisztráció vizsgálata:



21. ábra: Felhasználó regisztráció teszt

**Ez a teszt leírás az alábbiakat vizsgálja:**

Meg tud-e regisztrálni a felhasználó a <http://localhost:5173/register> címen elérhető felületen érvényes felhasználónévvel, e-mail címmel és jelszóval.

**Előkészületek:**

A teszt minden futtatása előtt meglátogatja a <http://localhost:5173/register> címet.

Vár 3 másodpercet, hogy a regisztrációs felület betöltődjön.

**Teszt esetek:**

**1. Regisztráció teszt:**

Beírjuk az Alma Repa felhasználónevet az #username beviteli mezőbe.

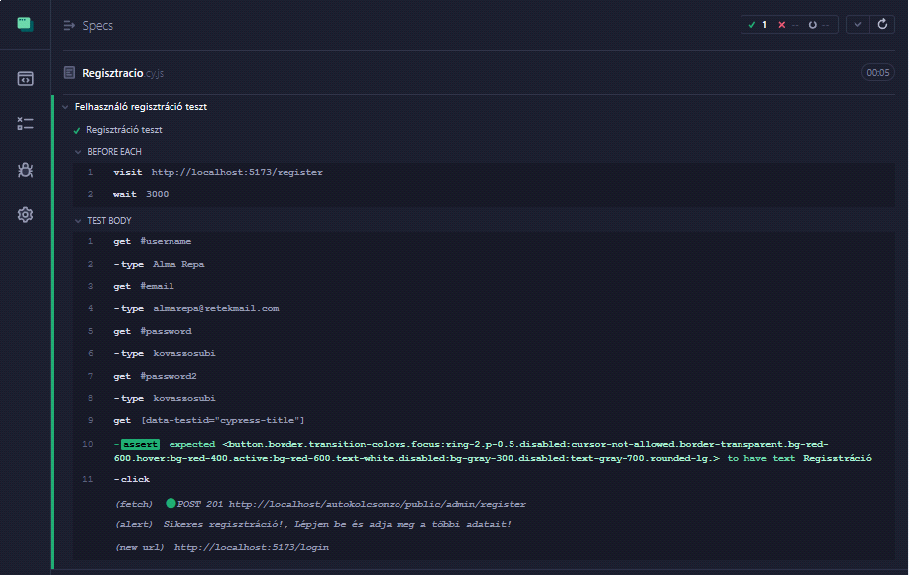
Beírjuk az almarepa@retekmail.com e-mail címet az #email beviteli mezőbe.

Beírjuk a kovaszosubi jelszót az #password beviteli mezőbe.

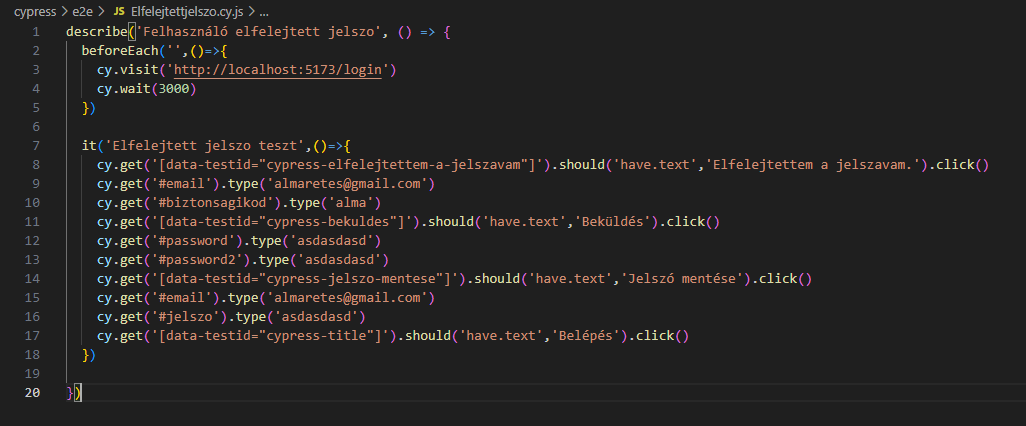
Beírjuk a kovaszosubi jelszót az #password2 beviteli mezőbe (a jelszó megerősítéséhez).

Kattintunk a "Regisztráció" gombra, amelynek data-testid attribútuma cypress-title.

Elvárás: A teszt sikeresnek minősül, ha a felhasználó sikeresen regisztrálódik, és átirányítják a bejelentkezési felületre.



22. ábra: Felhasználó regisztráció teszt eredménye

2.4.2.3.A sikeres jelszóváltoztatás vizsgálata:

23. ábra: Felhasználó elfelejtett jelszo teszt

Ez a teszt leírás az alábbiakat vizsgálja:

Meg tud-e változtatni a felhasználó a jelszavát a "Jelszó elfelejtése" funkció használatával, ha megadja a regisztrált e-mail címét és a biztonsági kódját.

**Előkészületek:**

A teszt minden futtatása előtt meglátogatja a <http://localhost:5173/login> címet.

Vár 3 másodpercet, hogy a bejelentkezési felület betöltődjön.

**Teszt esetek:**

**1. Elfelejtett jelszó teszt:**

Kattintunk az "Elfelejtettem a jelszavam." linkre, amelynek data-testid attribútuma cypress-elfelejtettem-a-jelszavam.

Beírjuk az almaretes@gmail.com e-mail címet az #email beviteli mezőbe.

Beírjuk az alma biztonsági kódot az #biztonsagikod beviteli mezőbe.

Kattintunk a "Beküldés" gombra, amelynek data-testid attribútuma cypress-bekuldes.

Beírjuk az asdasdasd jelszót az #password beviteli mezőbe.

Beírjuk az asdasdasd jelszót az #password2 beviteli mezőbe (a jelszó megerősítéséhez).

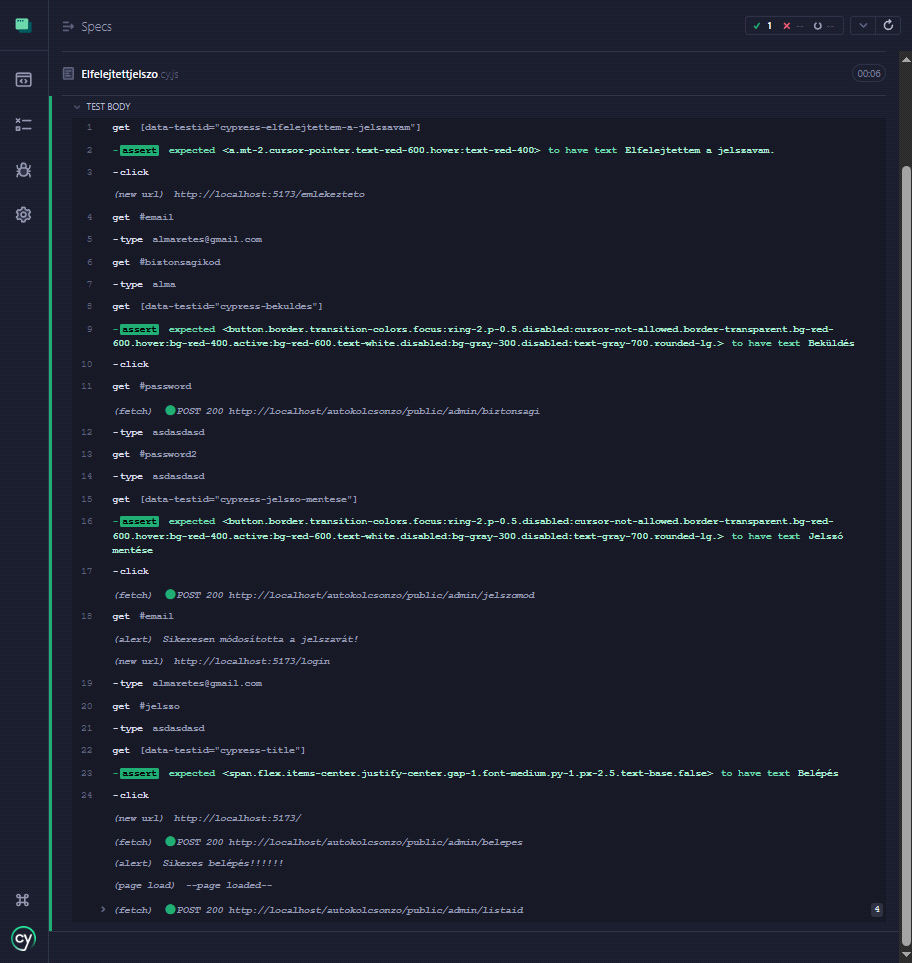
Kattintunk a "Jelszó mentése" gombra, amelynek data-testid attribútuma cypress-jelszo-mentese.

Beírjuk az almaretes@gmail.com e-mail címet az #email beviteli mezőbe.

Beírjuk az asdasdasd jelszót az #jelszo beviteli mezőbe.

Kattintunk a "Belépés" gombra, amelynek data-testid attribútuma cypress-title.

Elvárás: A teszt sikeresnek minősül, ha a felhasználó sikeresen be tud jelentkezni az új jelszóval.



24. ábra: Felhasználó elfelejtett jelszo teszt eredménye

2.4.2..A sikeres módosítás vizsgálata:



25. ábra: Módosítás teszt

**Ez a teszt leírás az alábbiakat vizsgálja:**

Meg tud-e módosítani a bejelentkezett felhasználó a profilját a "Módosítás" funkció használatával.

**Előkészületek:**

A teszt minden futtatása előtt meglátogatja a <http://localhost:5173/login> címet.

Beírja az almaretes@gmail.com e-mail címet és az asdasdasd jelszót.

Bejelentkezik a rendszerbe.

Vár 3 másodpercet, amíg a profil betöltődik.

Meglátogatja a <http://localhost:5173/modositas> címet.

*Teszt esetek:*

**1. Módosítás teszt:**

Beírja a "Nagy Géza" nevet az #nev beviteli mezőbe.

Beírja az "asdasd" biztonsági kódot az #biztonsagikod beviteli mezőbe.

Beírja a "123123123" jogosítvány számát az #jogositvanyszama beviteli mezőbe.

Beírja a "06706899874" telefonszámot az #telefonszam beviteli mezőbe.

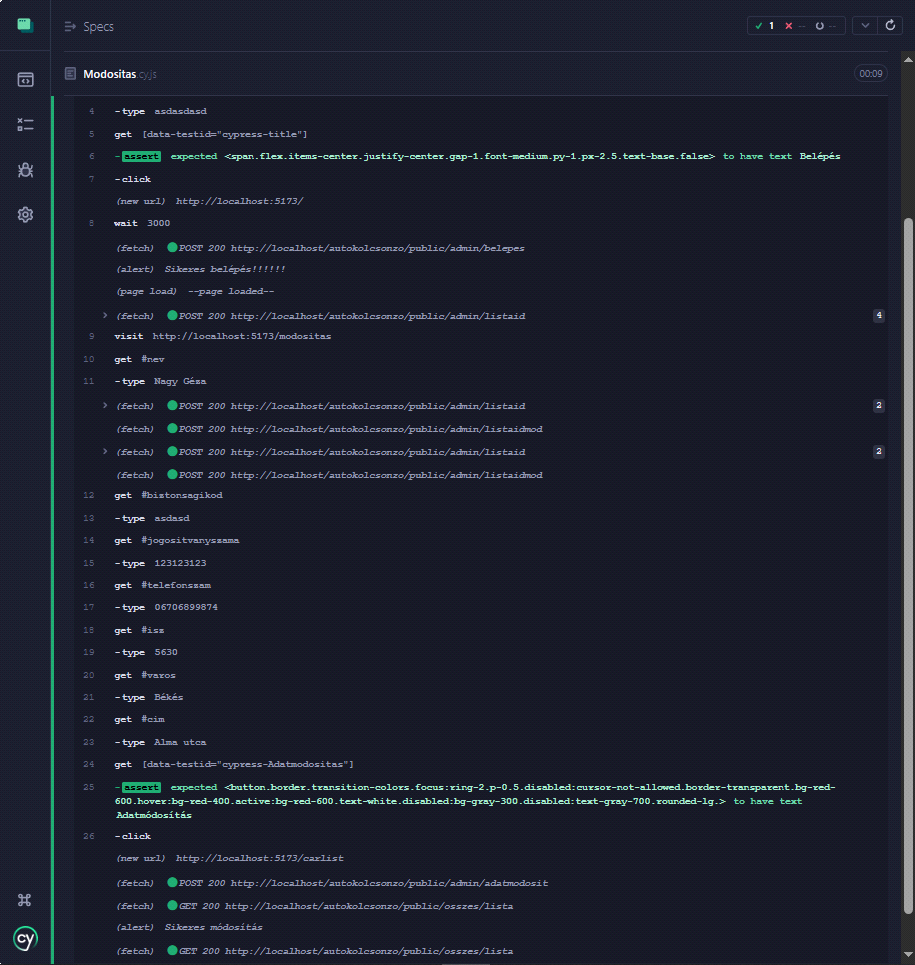
Beírja az "5630" Irányítószámot az #isz beviteli mezőbe.

Beírja a "Békés" várost az #varos beviteli mezőbe.

Beírja az "Alma utca" címet az #cim beviteli mezőbe.

Kattint az "Adatmódosítás" gombra, amelynek data-testid attribútuma cypress-Adatmodositas.

Elvárás: A teszt sikeresnek minősül, ha a felhasználó profilja frissül az új adatokkal.



25. ábra: Módosítás teszt eredménye

2.5. Fejlesztési lehetőségek:

A webapplikáció alapvető funkciói már működnek, de van lehetőség a felhasználói élmény javítására további bővítésekkel. Az alábbiakban javaslatokat fogalmazunk meg a weboldal felhasználóbarátabbá tételére:

1. Keresés és szűrés: Továbbfejlesztett keresési és szűrési funkciók bevezetése, amelyek sebességváltóját, felszereltségét és egyéb kritériumok alapján szűrjék a keresési eredményeket.

2. Foglalási naptár: Integrált foglalási naptár bevezetése, amely valós idejűen mutatja az elérhető autókat és a foglalási árakat.

3. Helyszín alapú keresés: Helyszín alapú keresés bevezetése, amely lehetővé teszi a felhasználók számára, hogy a kívánt helyszínen elérhető autókat keressék.

4. Árajánlatkérés: Árajánlatkérés funkció bevezetése, amely lehetővé teszi a felhasználók számára, hogy egyéni árajánlatot kérjenek a kívánt autóra és időszakra.

5. Automatizált fizetési rendszer: Automatizált fizetési rendszer bevezetése, amely lehetővé teszi a felhasználók számára, hogy online foglaljanak és fizessenek a kölcsönzésért.

6. Értékelési és véleményezési rendszer: Értékelési és véleményezési rendszer bevezetése, amely lehetővé teszi a felhasználók számára, hogy értékeljék a kölcsönzési élményt és megosszák véleményüket más felhasználókkal.

7. Mobilalkalmazás: Mobilalkalmazás fejlesztése, amely lehetővé teszi a felhasználók számára, hogy okostelefonjukon vagy táblagépükön foglaljanak autót, kezeljék a foglalásaikat és kövessék a kölcsönzési élményt.

8. Többnyelvű támogatás: Többnyelvű támogatás bevezetése a weboldalhoz és a mobilalkalmazáshoz, hogy a külföldi felhasználók is könnyen használhassák a szolgáltatást.

9. Átlátható árazás: Átlátható árazás biztosítása, amely tartalmazza az összes díjat és költséget, hogy a felhasználók pontosan tudják, mennyibe kerül a kölcsönzés.

10. Növelhetjük a weboldal biztonságát a legújabb biztonsági protokollok és technológiák bevezetésével. Ez megvédi a felhasználói adatokat a hackerektől és a más online fenyegetésektől. Beépíthetünk kéttényezős hitelesítést a weboldalra, hogy további védelmi szintet adjunk a felhasználói fiókokhoz.

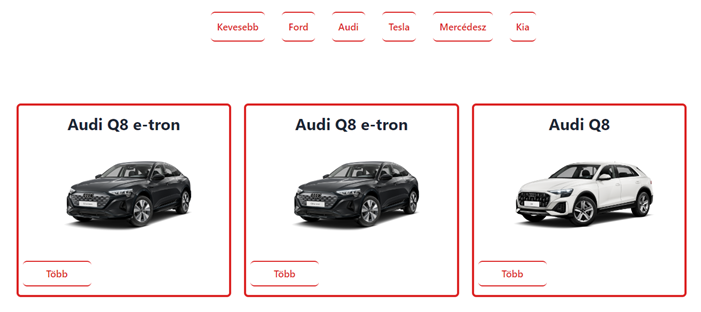
11. A Regisztrációhoz és a Jelszó emlékeztető funkciüt kibővíteni e-mail alkalmazásával. Regisztráció után illetve a jelszó módosításához elleneőrző e-mail küldése a felhasználóhoz.

# 3. Felhasználói dokumentáció

26. ábra: Az Autó Kölcsönzés főoldala

Az Autókölcsönző egy webes applikáció. A felhasználónak lehetősége van a bérelhető autókból szemezgetni. A autó foglalásához mindenképpen van szükség egy regisztációra. A belépett felhasználó tud így foglalni egy autót, látja a foglalásait és azokat módosíthatja és törölheti is.

## 3.1. Az alkalmazás felépítése és funkciói

A főoldalon nem bejelentkezett felhasználók négy menüpontot találnak.

26. ábra: Bérautó menü oldala

A Bérautók menüpontban lehet megtalálni a bérelhető autók képeit. Ha bővebb információra van szüksége, akkor a Több gombra kattintva bővebb információt talál. A fenti sávban pedig szűrheti az egyes márkák típusait. A Kevesebb gombra kattintva eltünteni a felsorolásból az adott márkát. Ha a Márka nevekre kattint, akkor tudja bővíteni a kocsik listáját. Az oldal bármelyik részén a felhasználó a *Főoldal* menüpontot választja, akkor mindig kezdőoldalra navigálja az applikáció.

### 3.1.1. Regisztráció menüpont

Az autó foglalásához regisztrációra van szükség. A regisztrációt a menüsor Regisztráció pontjával lehet megkezdeni. A regisztráció során meg kell adni a felhasználó nevét, E-mail címét illetve egy jelszót. A jelszót adatbiztonság miatt kétszer kell megadni, hogy ne lehessen véletlenül elírni. Ha a két jelszó nem egyezik meg, akkor a regisztráció nem lesz sikeres. Egy E-mail címmel csak egyszer lehet az oldalra regisztrálni.

Formai előírások:

A felhasználónév tartalmazhat ékezetes betűket, számokat, pontot, aláhúzást, kötőjelet bármilyen karakter.

Az email cím tartalmazhat betűket, számokat, pontot, kötőjelet, aláhúzást; @ és . karaktert tartalmaznia kell az email címnek

27. ábra: Regisztráció űrlapja

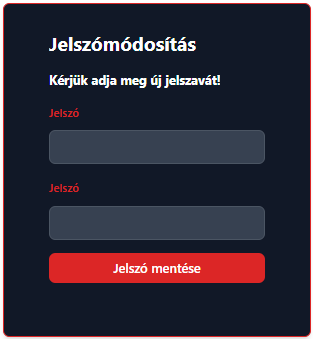
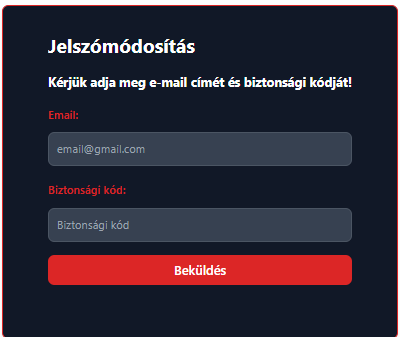
A két jelszó mezőben megadott értékeknek egyeznie kell.

A Regisztráció gombra kattintva lehet rögzíteni az új felhasználót. Sikeres regisztráció után a Sikeres regisztráció! Lépjen be és adja meg a többi adatait! üzenet jelenik meg.

Ezek után a Belépés oldalra navigál az oldal és ott lehet megadni a korábban regisztrált jelszó-e-mail cím párost. Ha nem egyezik meg a regisztráció során leadott adatokkal a belépés nem lesz sikeres. A sikeres belépésről a rendszer tájékoztató üzenetet küld.

Az Elfelejtettem a jelszavam funkcióval lehet új jelszót beállítani. Ennek egy fontos feltétele van, hogy a felhasználó megadta korábban a Biztonsági kódját.

A jelszómódosítás menete:

Az Elfelejtettem a jelszavamra kattinttással kezdődik. A módosítás két lépcsőből áll. Először meg kell adni az e-mail címet és a biztonásgi kódot. A Beküldés gombra kattintva a rendszer a regisztrációnál megadott adatokkal ellenőrzi és ha megegyezk akkor megjelenik, egy másik űrlap ahol megkell adni az új jelszót kétszer azonosan. A jelszó mentése gombra kattintva, ha a két jelszó mezőbe írt adat megegyezik, akkor sikeresen módosította a felhasználó a jelszavát, amiről üzenetet is kap. Ettől kezdve már csak az új jelszóval kell belépni az oldalra.

28. ábra: Jelszó módosítás űrlapjai

Belépés után a menüsor átalakul. Eltűnik a Regisztrávió menüpont és helyette új pontok jelennek meg.

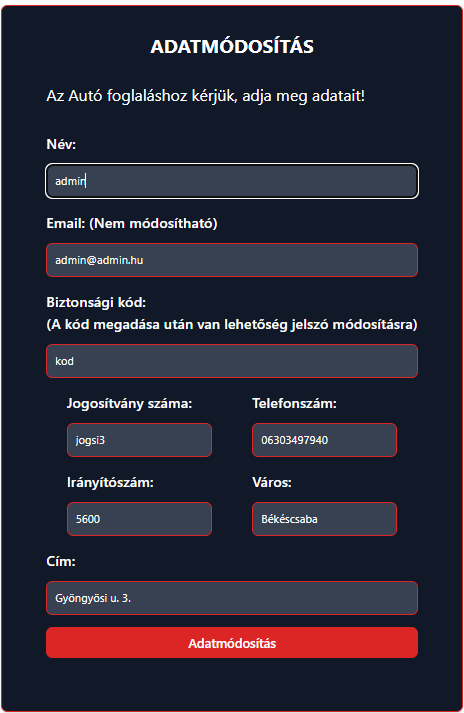
* Adatmódosítás menüpontban a felhasználó adatait tudja módosítani illetve kibővíteni.

29. ábra: Belépés utáni Menüsor

* A Rendeléseim menüpontban találja meg azokat az autó foglalásokat, amiket már korábban a felhasználó lefoglalt. Ott tudja módosítani és törölni őket.
* A Kilépés gombra kattintva a felhasználó kilép az oldalról.
* A Kilépés gomb mellett található a belépett felhasználó neve. A névre kattintva az adatait tartalmazó űrlap jelenik meg.

A belépés után célszerű az adatmódosítással kezdeni és még további adatokat megadni.

### 3.1.2. Adatmódosítás menüpont

Az adatmódosításkor az e-mail címet nem lehet módosítani.

A módosítható mezők:

* Név
* Biztonsági kód: Jelszó módosításnál van jelentősége
* Jogosítvány száma
* Telefonszáma
* Irányítószám\_ 4 karakter hosszó lehet
* Város
* Cím

Adatmódosítás gombra kattintva elnavigálunk a Bérautók menüpontra illetve egy Sikeres módosítás üzenetet kapunk, az adatainkat módosítottuk.

A Kilépés gomb mellett található a felhasználó neve. Ha erre a gombra kattintunk, akkor az Adatmódosítás ürlaphoz hasonló jelenik meg, de itt az adatok csak tájékoztató jellegűek, mert itt nem lehet azokat módosítani csak akkor, ha az Adatmódosítás gombra kattintunk.

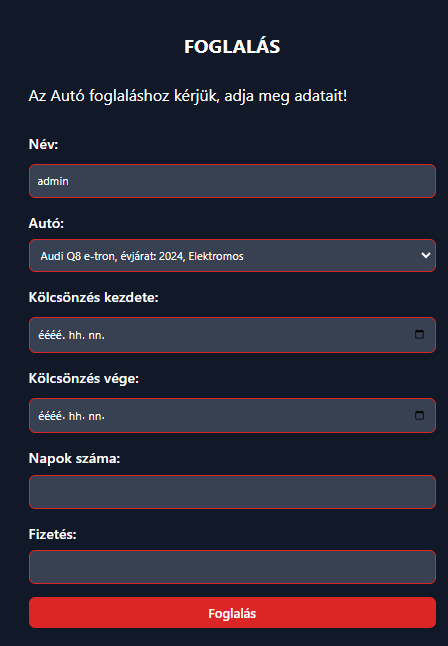
29. ábra: Adatmódosítás űrlapja

### 3.1.3. Foglalás menüpont

Az autót csak bejelentkezett felhasználó foglalhat.

Foglalásnál menete:

A fenti menüsoron a Foglalás menüpontra kattintunk. Ez után a Foglalás oldalra naigálunk. Az oldal az alábbi adatokat tartalmazza:

* *Név:* Nem kell megadni. Autómatikusan kitölti a rendszer a bejelentkezés alapján
* *Autó:* Egy választó menüból lehet kiválasztani a foglalni kívánt autót. A rendszer figyeli, hogy az autó az adott időtartalmon szabad e vagy nem. Ha már kikölcsönözték akkor nem kölcsönözhető.
* *Kölcsönzés kezdete:* Dátum mező. Az első napját kell megadni amitól szeretnénk használni az autót.
* *Kölcsönzés vége:* Dátum mező. A kölcsönzés utolsó napját kell kiválasztani. Természetesen nem lehet korábbi mint a Kölcsönzés kezdete.
* *Napok száma:* Számított mező. Autómatikusan számolja ki a rendszer a Kölcsönzés kezdete és vége dátum mezőkből. Nem lehet negatív szám.
* *Fizetés:* Szintén számított mező. A Napok számát szorozza meg 10000-el.

30. ábra: Foglalás űrlapja

A Foglalás gomb nyomásával megvizsgálja az adatokat a rendszer, és ha mindent rendben talál akkor egy Sikeres foglalás üzenettel betölti a Rendeléseim oldalt.

### 3.1.4. Rendeléseim menüpont

30. ábra: Rendeléseim táblázat

Az oldal táblázatos elrendezésben mutatja meg a felhasználó rendeléseit. Kibővítve az Autó pontos nevével és képével. A rendeléseket módosítani lehet a Módosítáés gombbal. Ha rákattintunk a gombra akkor kiválasztott rendelés adataival betölti az Foglalás módosítása oldalt ami ugyanúgy néz ki mint a Foglalás oldal. A módosítás mentése után a rendszer visszanavigál a Rendeléseim oldalra.

Ha a Törlés gombra kattintunk akkor akkor a rendszer a kiválasztott rendelést törli az adatbázisból.

# 4. Összegzés

Amikor elkezdtük életünk első web applikációját készíteni nem is sejtettük, hogy milyen nehézségekbe ütközünk. De megállapíthatjuk, hogy az Autó kölcsönző webapplikációnk a tervezett kezdeti elvárásainknak megfelelően került elkészítésre. Természetesen nem csak nehézségekbe ütköztünk, hanem voltak boldog pillanatok is.. Amikor először sikerült egy általunk létrehozott végponttal az első „fetch”, vagy amikor megláttuk, hogy az elgondolt oldal hogy fog kinézni illetve az az érzés, amikor úgy éreztük, hogy készen vagyunk. Természetesen ilyenkor jönnek elő, hogy még ezt meg azt kéne bele rakni, de már közeleg a leadási határidő. Az oldal elkészítése során az alábbi tapasztalatokkal gyarapodtunk.

A projektünk első fázisa a tervezés sokáig tartott. A szoftverfejlesztő párosok kijelölése nem ment könnyedén elég későre alakultak ki a párok. Az autó kölcsönő öteltét szinte egyszerre találtuk ki. Onnan már csak a funkciók meghatározása és az adatbázis tervezése volt a legfontosabb. A funkciók tekintetében fontos volt, hogy lehessen autót foglalni illetve legyen adminisztráció is benne, regisztráció, módosítás és törlés is. A fő 3 adatbázis táblát hamar kitaláltuk onnen már csak a végpontok meghatározásán volt a figyelmünk. Ezen időszakban a konzultációink chatten illetve az oktatási napokon voltak. Mivel mindketten dolgozunk a képzés mellett így csak a munka után illetve hétvégente tudtunk az applikációval foglalkozni. A oktatási napok után nehéz volt közös időt találni a konzultációra hiszen nem egy műszakban dolgozunk. Így sokszor csak chatten üzentünk egymásnak, és amikor a másik fél felkelt akkor tudott reagáli rá. A projekt készítés közben tapasztatuk meg hogy milyen egy szoftverfejlesztő élete illetve milyen egy szoftver készítése csapatban. A tapasztalatunk az, hogy sok egyeztetést, megbeszélést igényel, legalábbis kezdő szinten biztosan. Azért elmondhatjuk hogy amit célként megfogalmaztunk az applikációval kapcsolatban azokat a célokat nagy részt teljesítettük. Persze a munka közben sokszor eszünkbe jutott hogy de jó lenne még megcsinálni valamit de az idő nagy úr. De kijelenthetjük a projektünket sikeresen befejeztük. Nyilván vannak benne hibák de egy szoftver sosincs kész. Mindig változik mindig javul.

Szeretnénk megköszönni a tanárainknak, hogy a délutánjaikat, estéjüket néha a szombatjaikat ránk szánták. Amikor valamit nem értettünk, akkor mindig készséggel segítettek, ha elakadtunk valamikor akkor segítettek. Továbbá szeretnénk köszönetet mondani családtagjainknak, szeretteinknek, kollegáinknak akik készséggel álltak mellettünk és elnézték nekünk, hogy sok esti órát a gép előtt töltöttünk vagy netán ha munka közben is a C# , Java vagy a REACT szépségeiben gyönyörködtünk.

# 5. Irodalomjegyzék

A vizsgaremekünk elkészítése során az alábbi segédanyagokat használtuk fel:

* <https://hu.wikipedia.org/wiki/Visual_Studio_Code>
* <https://hu.wikipedia.org/wiki/XAMPP>
* <https://hu.legacy.reactjs.org/docs/getting-started.html>
* <https://blog.juszkocs.hu/2023/08/tailwind-masik-legjobb-css-keretrendszer.html>
* <http://ade.web.elte.hu/w/CodeIgniter_PHP_Framework.pdf>
* <https://www.youtube.com/watch?v=H4bS5Zp6nUk&list=PLt2fZkYs6q_l2WebLGr6biyk551rLUtLV&index=5>
* <https://tailwindflex.com/>
* <https://tailwindcss.com/docs>
* <https://gemini.google.com/app>
* <https://chatgpt.com/?oai-dm=1>
* <https://www.youtube.com/playlist?list=PL4cUxeGkcC9jZIVqmy_QhfQdi6mzQvJnT>