

Motorkölcsönző szolgáltatás

vizsgaremek

2025.

Készítette:

Bai Máté

Kádasi Emese Tímea

Szabó Bálint

Tartalom

1.	A szolgáltatás általános bemutatása.....	4
1.1.	A motorkölcsönző.....	4
1.	Fejlesztői dokumentáció	5
1.1.1	A szerver oldal bemutatása	5
1.1.2	A szerver oldali alkalmazás.....	5
1.1.3	Adatbázis.....	5
1.2	A felhasználói weboldalak bemutatása.....	9
1.2.1	Kölcsönzés működése	9
1.2.2	Regisztráció	9
1.2.3	Megerősítés	9
1.2.4	Bejelentkezés	9
1.2.5	Profil.....	10
1.2.6	A működés.....	10
1.3	A mobilapplikáció bemutatása	25
1.3.1	Működése.....	25
1.3.2	Bejelentkezés	26
1.3.3	Kijelentkezés	27
1.3.4	Profilbetöltés	28
1.4	Adminisztrációs alkalmazás bemutatása	31
1.4.1	Adminisztrátori alkalmazás	31
1.4.2	Szuperadminok oldalai.....	34
1.4.3	Szerelők oldalai	39
1.4.4	Recepziósok oldalai.....	45
2.	Felhasználói dokumentáció	51
2.1	A felhasználói weboldalak bemutatása.....	51
2.1.1	Felhasználói adatok	51
2.1.2	Felhasználói felület	51
2.1.3	Foglalás menete	52
2.2	A mobilapplikáció bemutatása	53
2.2.1	Mobil application	53
2.3	Adminisztrációs alkalmazás bemutatása	54
2.3.1	Adminisztrátori alkalmazás	54
2.3.2	Szuperadmin felület	55
2.3.3	Szerelő felület.....	58

2.3.4	Recepciós felület	62
3.	Tesztelési dokumentáció	67
3.1	Szerveroldal tesztelése	67
3.1.1	API Teszt.....	67
3.2	A felhasználói weboldalak tesztelése.....	68
3.2.1	Regisztráció	68
3.2.2	Bérlet hozzáadása	69
3.2.3	Felhasználó profil módosítás.....	70
3.2.4	Eszköz hozzáadása a foglaláshoz.....	71
3.2.5	Eszköz törlése a foglalásból.....	72
3.2.6	Foglalás törlése	73

1. A szolgáltatás általános bemutatása

Egy olyan szolgáltatást hoztunk létre, ami támogatja egy motorkölcsönző működését. A szolgáltatás révén az ügyfelek online módon válogathatnak és foglalhatnak le az elérhető motorok közül. A motor kölcsönző pedig a szoftver segítségével kezeli és tárolja a motorok és foglalások adatait.

A szolgáltatást az ügyfelek egy weboldalon keresztül érhetik el, valamint a fontosabb funkciók (élő foglalások megtekintése, törlése) mobilra optimalizált változatában is elérhetőek. A motorkölcsönző alkalmazottai pedig a kölcsönzéssel kapcsolatos adatok kezelésére szolgáló felületeket egy kliens alkalmazásban érik el.

1.1. A motorkölcsönző

A weboldalon rengeteg motorból tudnak válogatni az ügyfelek, ami folyamatosan cserélünk és bővíünk.

A motorkölcsönző szolgáltatást úgy képzeltük el, hogy minden kerületben van egy-egy telephelyünk, ahol vannak a mi motorjaink. A telephelyről lehet elvinni a motorokat és valamelyik másik vagy ugyanarra telephelyünkre visszavinni.

Minden telephelyen vannak recepciók, akiknek a feladata, hogy kiadják a motorokat és a kért felszereléseket az adott méretben (protektoros ruha, cipő, sisak) az ügyfeleknek.

Jelenleg csak Budapesten vannak a telephelyeink, de jövőben szeretnénk majd terjeszkedni, illetve egy kerületen belüli telephely bővítésére.

Motorok

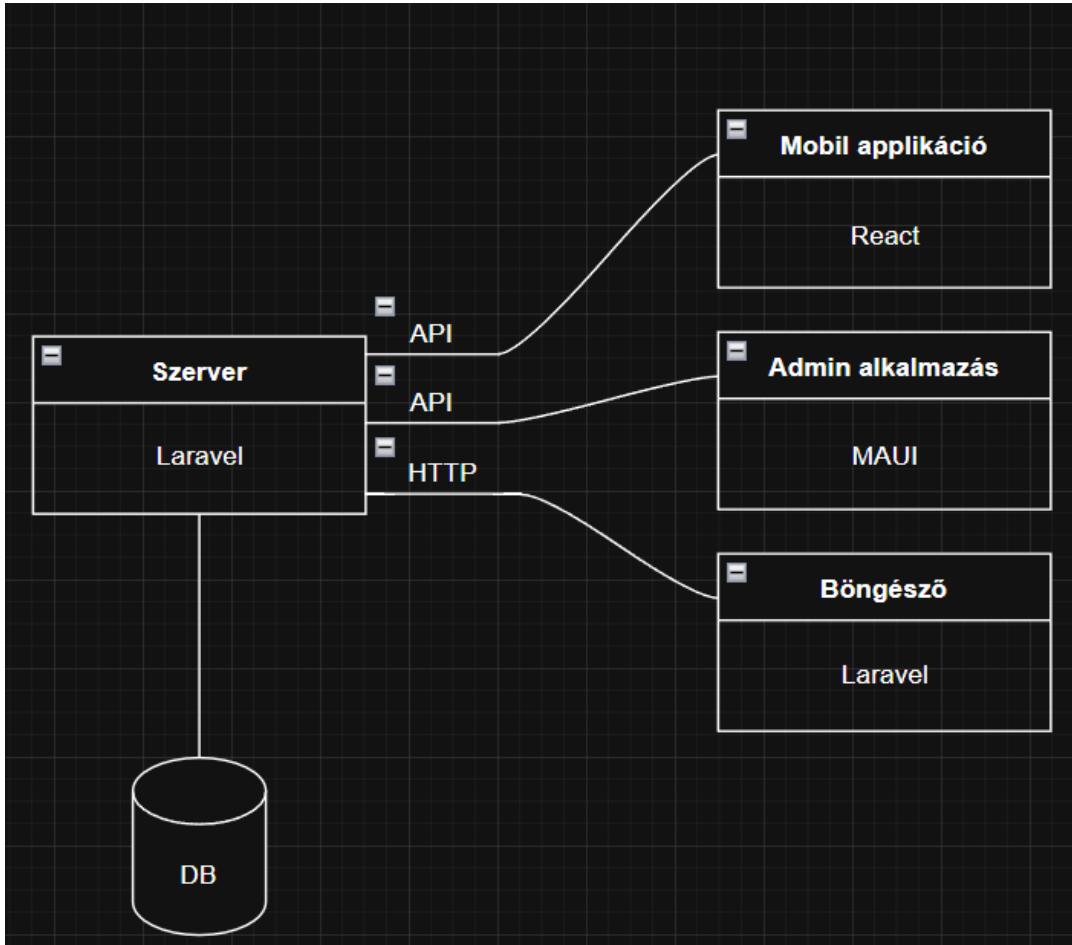
Az ügyfél tud szűrni **márkára**, **elhelyezésre**, **korra**, **férőhelyre** (egy v. két férőhelyes), **sebességváltó típusra** (automata, manuális) és **meghajtás típusára** (elektromos, benzin), illetve kölcsönzési intervallumra is.

A felhasználó legalább 80%-os tank töltöttséggel kapja meg a motorját és minimum annyival kell leadnia. Máskülönben, leadásnál plusz költséget számolunk fel.

Ha a motor sérült állapotban kerül leadásra akkor ezt a recepció feljegyzi és levonja a kaucióból.

1. Fejlesztői dokumentáció

A motorkölcsönző szolgáltatás architektúrája:



1.1.1 A szerver oldal bemutatása

Az alábbiakban bemutatjuk a szerver oldal működését.

1.1.2 A szerver oldali alkalmazás

A szerver oldal laravelben készült. A mobil applikáció és az admin alkalmazás API-n keresztül éri el, míg a felhasználói felületet http protokollon keresztül teszi elérhetővé a felhasználók böngészői számára. A szerver oldalon egy MySQL adatbázis fut.

1.1.3 Adatbázis

motorcycles table

brand	A motor márka (szöveg, maximum 20 karakter)
type	A motor típusa (szöveg, maximum 100 karakter)

licencePlate	A motor rendszáma (szöveg, maximum 7 karakter, egyedi)
year	A motor évjárata (szám, 0 – 65.535)
gearbox	A motor sebeségváltó típusa (szöveg, maximum 25 karakter)
fuel	A motor üzemanyag típusa (szöveg, maximum 1 karakter)
powerLe	A motor lóereje (törtszám)
powerkW	A motor kilowattja (törtszám)
engineSize	A motor motorjának mérete (törtszám)
drivingLicence	A motort milyen jogosítvánnyal vezethető (szöveg, maximum 4 karakter)
places	A motor férőhelye (szám, 0 - 255)
price	A motor bérlesi összege (szám, 0 - 4294967295)
deposit	A motor kaució összege (szám, 0 - 4294967295)
trafficDate	A motor forgalmijának lejáratí ideje (dátum)
location	A motor aktuális helyszíne (szöveg, lehet null értéke)
image	A motor képének a neve (szöveg)
isInService	A motor szervizben van-e (logikai érték)
problemComment	A motor problémái (szöveg, lehet null értéke)

loans table

rentalDate	A bérző bérlesének kezdeti dátuma (dátum)
returnDate	A bérző bérlesének visszahozási dátuma (dátum)
gaveDown	A motort leadták-e (logikai érték)
problemDescription	A Userrel vagy a motorral adott problémák (szöveg)
orders_id	Minden egyes rendelésnek egyedi azonosítója (szöveg)
motorcycle_id	Az adott kölcsönzéshez tartozó motorcycles azonosító (szám, egyedi)

users_id	Az adott kölcsönzéshez tartozó users azonosító (szám, egyedi)
----------	---

tools table

toolName	Az eszköz/felszerelés neve (szöveg, maximum 80 karakter)
size	Az eszköz/felszerelés mérete (szöveg, maximum 255 karakter)

device switches table

tools_id	Az adott eszközhöz tartozó eszköz/felszerelés azonosító (szám, egyedi)
loans_id	Az adott eszközhöz tartozó kölcsönzési azonosító (szám, egyedi)

admin table

name	Az admin neve (szöveg, maximum 255 karakter)
email	Az admin email címe (szöveg, maximum 255 karakter, egyedi)
password	Az admin jelszava (szöveg, maximum 255 karakter)
jobStatus	Az admin munkaköri státusza (szám, maximum 255 karakter)

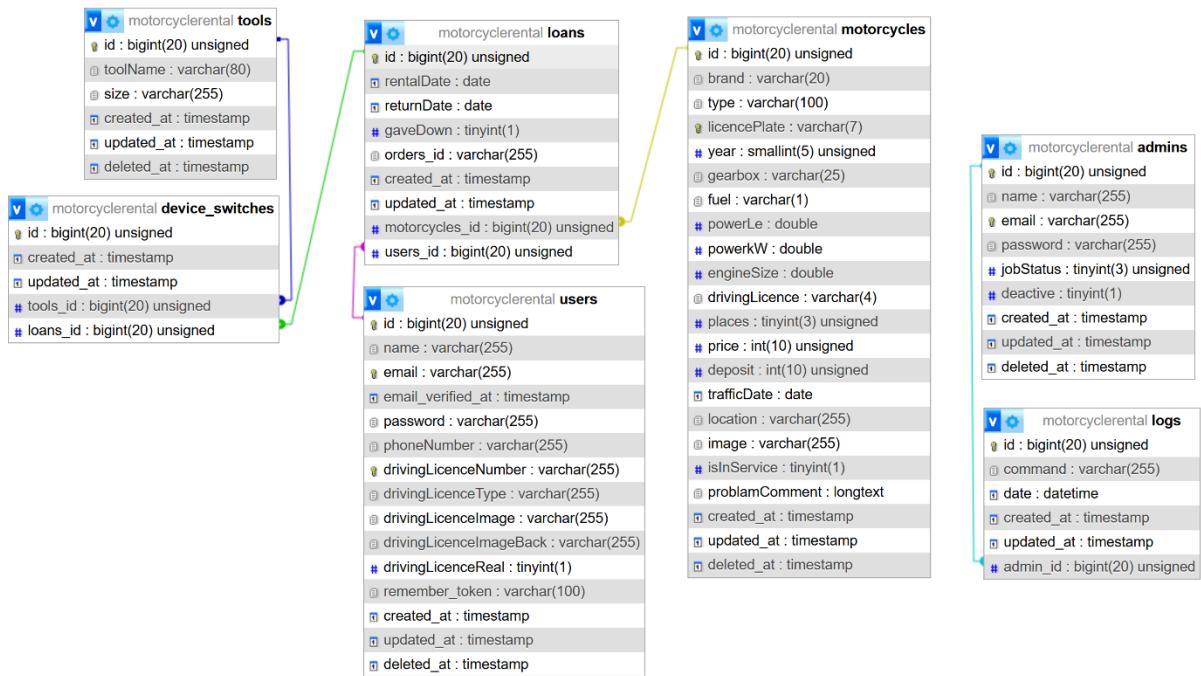
users table

name	A felhasználó neve (szöveg, maximum 255 karakter)
email	A felhasználó email címe (szöveg, maximum 255 karakter, egyedi)
password	A felhasználó jelszava (szöveg, maximum 255 karakter)
phoneNumber	A felhasználó telefonszáma (szöveg, maximum 255 karakter)
drivingLicenceNumber	A felhasználó jogosítvány egyedi azonosítója (szöveg, maximum 255 karakter, egyedi)

drivingLicenceType	A felhasználó jogosítvány típusa (szöveg, maximum 255 karakter)
drivingLicenceImage	A felhasználó jogosítvány képének a neve (az előlapi oldala) (szöveg, maximum 255 karakter)
drivingLicenceImageBack	A felhasználó jogosítvány képének a neve (az hátoldali oldala) (szöveg, maximum 255 karakter)
drivingLicenceReal	A felhasználó jogosítványa valódi és érvényes-e (logikai érték)

logs table

command	A parancs leírása (szöveg, maximum 255 karakter)
date	A parancs végrehajtási dátuma (dátum)
admin_id	Az adott loghoz tartozó admin azonosító (szám, egyedi)



1.2 A felhasználói weboldalak bemutatása

1.2.1 Kölcsönzés működése

A felhasználók egy általunk megosztott szerveren futó weboldalra tudnak fellépni, amelyen a motorokat tudják kibérelni. Ezeket a motorokat kívánt ELÉRHETŐ napokon tudják kibérelni, személy szerint kívánt eszközökkel, melyeket rendelés előtt egy nappal tudnak törölni a vagy hozzáadni a felhasználó. minden eszköz típusból legfeljebb 2 db-ot lehet kölcsönözni. Az eszközökért egyedi kauciót számolunk fel a típustól függően.

Kedvezményekkel is kínáljuk a felhasználókat. Ha 3 vagy annál több napra kölcsönöz a személy, akkor **-20%** kedvezményt vonunk le a napi árból, 7 vagy annál több napnál meg már **-30%** a járó kedvezmény.

1.2.2 Regisztráció

A felhasználó amíg nem regisztrált be addig nem tud bármilyen jelentős aktivitást végezni az oldalon(bérlést). Ezért el kell végeznie a regisztrációt, ha szeretne bérelni az oldalunkon.

Meg kell adnia a teljes nevét, email címét, telefonszámát, jogosítvány azonosítóját, jogosítvány típusát és olvasható kép formátumban a jogosítványnak elejét és hátulját. Ezen felül egy egyedi jelszóval is kell rendelkeznie, amit saját maga ad meg.

A jelszónak legalább 8 karakter hosszúnak kell lennie, tartalmaznia kell kis és nagybetűt, és legalább egy számot.

1.2.3 Megerősítés

Regisztrálást követően küldünk egy a felhasználó által megadott elérhetőségi email címre, amellyel megerősíti, hogy tényleg övé az email cím vagy hogy létezik-e.

Ezen felül az adminisztrátornak meg kell erősítenie, hogy valós, és aktív a jogosítvány, amelyet feltöltött a felhasználó.

Amíg a fenti 2 feltétel nem valósul meg, addig a felhasználó képtelen bármilyen aktivitást végezni(bérlést).

1.2.4 Bejelentkezés

A bejelentkezéskor a felhasználó az email címével és a jelszavával tud belépni a profiljába, ha a felhasználó elfelejtette a jelszavát, akkor az **“Elfelejtett jelszó”** mezőre kattintva az adott email címre küldünk egy emailt, amelyre kattintva be tudja állítani az új jelszavát.

Bejelentkezés után a felhasználó bármilyen aktivitást végezhet, ha az összes regisztrációs feltételeket teljesítette. Tud új motort bérelni eszközöt hozzáadni vagy törölni a bérleshez és törölni a rendelést, ha megadott időintervallumon kívül esik a kívánt művelet.

1.2.5 Profil

Bejelentkezés után a felhasználó megtudja tekinteni és módosítani a profilját. A megnevezését, elérhetőségeit, és a jogosítványról készült képeit, ha esetleg újat csináltat a felhasználó. Az utóbbi esetben újabb ellenőrzésen megy keresztül a jogosítványa.

Ezen felül itt tudja a rendelési előzményeit megtekinteni, azokat törölni vagy módosítani.

1.2.6 A működés

1.2.6.1 Regisztráció

Az oldalra lépve a felhasználónak regisztrálnia kell foglalás érdekében. Amelyet a jobb oldalt a lenyíló menüben a regisztráció gombra nyomva tud megtenni.

Az adatok megadása során a regisztrációs gombra nyomva feltöltyük az adatbázisba a felhasználó által megadott adatokat, ezzel regisztrálva őt.

1.2.6.2 Fájlok ellenőrzése és feltöltése

- Megvizsgáljuk, hogy a `drivingLicenceImage` és `drivingLicenceImageBack` mezők tartalmaznak-e feltöltött fájlokat.
- Ha igen, a fájlokat a `store` metódussal a `public/driving_licence_images` mappába mentjük, és elmentjük az elérési útjukat.

```
//Ellenőrizzük, hogy van-e feltöltött fájl  
if (isset($data['drivingLicenceImage']) && isset($data['drivingLicenceImageBack'])) {  
    //A fájl feltöltése és a fájl elérési útjának mentése  
    $filePath = $data['drivingLicenceImage']->store('driving_licence_images', 'public');  
    $filePathBack = $data['drivingLicenceImageBack']->store('driving_licence_images', 'public');  
}
```

1.2.6.3 Új felhasználó létrehozása

- A `User::create` metódus segítségével létrehozunk egy új felhasználót az adatbázisban, ahol:
 - A `name`, `email`, `phoneNumber`, `drivingLicenceNumber`, és `drivingLicenceType` mezők a kapott adatokból töltődnek.
 - A `password` biztonságosan, hash-elve kerül mentésre a `Hash::make` használatával.
 - A feltöltött fájlok elérési útját (`drivingLicenceImage`, `drivingLicenceImageBack`)

```

return User::create(attributes: [
    'name' => $data['name'],
    'email' => $data['email'],
    'password' => Hash::make(value: $data['password']),
    'phoneNumber' => $data['phoneNumber'],
    'drivingLicenceNumber' => $data['drivingLicenceNumber'],
    'drivingLicenceType' => $data['drivingLicenceType'],
    'drivingLicenceImage' => isset($filePath) ? $filePath : null,
    'drivingLicenceImageBack' => isset($filePathBack) ? $filePathBack : null,
]);

```

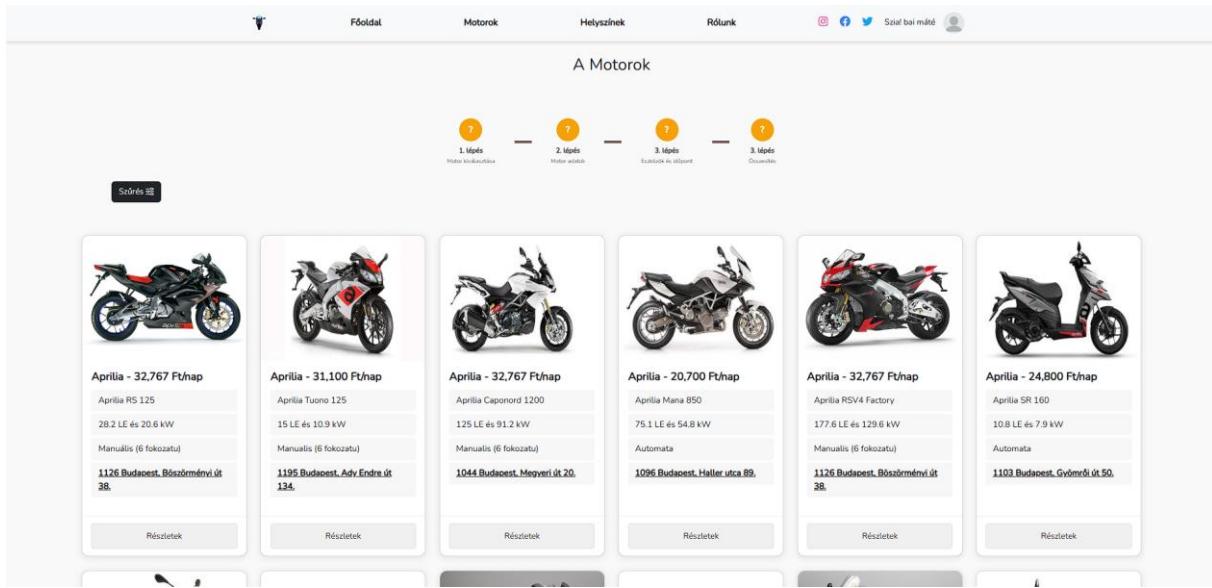
name	"John Doe"	Yes	User's full name.
email	"john@example.com" "	Yes	Unique email address.
password	"secret123"	Yes	Hashed via Hash::make().
phoneNumber	"123456789"	Yes	Contact number.
drivingLicenceNumber	"AB123456"	Yes	Driver's license ID.
drivingLicenceType	"B"	Yes	License category (e.g., B, C).
drivingLicenceImage	UploadedFile (front side)	Yes	Stored in public/driving_licence_images.
drivingLicenceImageBack	UploadedFile (back side)	Yes	Stored in public/driving_licence_images.

Ezt követően, korábban említve az email címet meg kell erősítenie a felhasználónak. Ehhez egy beépített laraveles metódust használtunk, amely elküldi az adott művelethez szükséges email-t.

1.2.6.4 Bérlés

Fent megemlítve, a navbar-on(a fenti menüben) lehet navigálni az oldalak között.

A **motorok** fülre kattintva ezt az oldal látjuk:



Itt felsorolja az összes motort, ami létezik a rendszerünkben. Amelynek az elérési útvonala így van implementálva:

```
Route::get(uri: "/motors", action: [MotorcycleController::class, "index"])->name(name: "motors.index");
```

A motorok adatai lekérése az adatbázisból:

```
$query = DB::table(table: 'motorcycles')
    ->leftJoin(table: 'loans', first: function ($join): void {
        $join->on('motorcycles.id', '=', 'loans.motorcycles_id')
        ->whereRaw('loans.id IN (SELECT MAX(id) FROM loans GROUP BY motorcycles_id)');
    })
    ->select(columns: 'motorcycles.*', 'loans.rentalDate', 'loans.returnDate')
    ->where(column: 'motorcycles.isInService', operator: 0);
```

Sorrendbe helyezi a megjelenített motorokat és csak azokat jeleníti meg amelyek nincsenek szervízi munkálatok alatt.

Ha az oldalon a bal oldali szűrés gombra kattint a felhasználó akkor tud szűrni, hogy milyen motort szeretne látni. Ez a korábban részletezve is volt, hogy milyen feltételekkel tud.

1.2.6.5 A szűrés működése

```

$(query = DB::table(table: 'motorcycles')
    ->leftJoin(table: 'loans', first: function ($join): void {
        $join->on('motorcycles.id', '=', 'loans.motorcycles_id')
        ->whereRaw('loans.id IN (SELECT MAX(id) FROM loans GROUP BY motorcycles_id)');
    })
    ->select(columns: 'motorcycles.*', 'loans.rentalDate', 'loans.returnDate')
    ->where(column: 'motorcycles.isInService', operator: 0);

    // Feltételek hozzáadása a többi szűrőhöz (brand, year, gearbox, fuel, location)
    if ($request->filled(key: 'brand')) {
        $query->where(column: 'brand', operator: $request->input(key: 'brand'));
    }
    if ($request->filled(key: 'year')) {
        $query->where(column: 'year', operator: $request->input(key: 'year'));
    }
    if ($request->filled(key: 'gearbox')) {
        $query->where(column: 'gearbox', operator: $request->input(key: 'gearbox'));
    }
    if ($request->filled(key: 'fuel')) {
        $query->where(column: 'fuel', operator: $request->input(key: 'fuel'));
    }
    if ($request->filled(key: 'location')) {
        $query->where(column: 'location', operator: $request->input(key: 'location'));
    }

    if ($request->filled(key: 'dateStart') && $request->filled(key: 'dateEnd')) {
        $dateStart = Carbon::parse(time: $request->input(key: 'dateStart'));
        $dateEnd = Carbon::parse(time: $request->input(key: 'dateEnd'));

        $query->where(column: function ($subQuery) use ($dateStart, $dateEnd): void {
            $subQuery->whereNull('loans.rentalDate')
            ->orWhere('loans.returnDate', '<', $dateStart)
            ->orWhere('loans.rentalDate', '>', $dateEnd);
        });
    }

    // Lekérdezés végrehajtása
    $motorcycles = $query->distinct()->get();

    // Kiegészítő lekérdezések szűrközhöz (brands, locations, years, gearboxes)
    $brands = DB::table(table: 'motorcycles')->select(columns: 'brand')->distinct()->get();
    $locations = DB::table(table: 'motorcycles')->select(columns: 'location')->distinct()->get();
    $years = DB::table(table: 'motorcycles')->select(columns: 'year')->distinct()->orderBy(column: 'year', direction: 'asc')->get();
    $gearboxes = DB::table(table: 'motorcycles')
        ->select(columns: DB::raw(value: "Automata" AS gearbox"))
        ->union(
            | query: DB::table(table: 'motorcycles')->select(columns: 'gearbox')->distinct()->where(column: 'gearbox', operator: '!=', val: 'automata')
        )
        ->get();

    // Motorok átadása a nézetnek
    return view(view: 'motors.index', data: compact(var_name: 'motorcycles', var_names: 'brands', 'locations', 'years', 'gearboxes'));
}

```

- Brand szűrés:** Ellenőrizzük, hogy a lekérdezés tartalmazza-e a márkat (**brand**). Ha igen, a lekérdezést szűrjük a megadott márkkára.
- Évjárat szűrés:** Ha az évjárat (**year**) meg van adva, csak az adott dátumban elérhető motorokat kérdezzük le.
- Sebességváltó szűrés:** Ha a sebességváltó típusa (**gearbox**) meg van adva, a találatokat az alapján szűkítjük.
- Üzemanyag szűrés:** Ha az üzemanyag típusa (**fuel**) szerepel a lekérdezésben, annak megfelelően szűrjük az adatokat.
- Helyszín szűrés:** Ellenőrizzük, hogy a helyszín (**location**) mező ki van-e töltve, és csak az adott helyszínhez tartozó adatokat adjuk vissza.
- Időszak szűrés:** Ha kezdő és záró dátum (**dateStart** és **dateEnd**) is meg van adva, ellenőrizzük, hogy az adott időszak alatt a jármű elérhető-e. Ez kizára azokat, amelyek már foglalva vannak a megadott időszakra.
- Egyedi eredmények lekérdezése:** A lekérdezés végén az eredményt szűrjük, hogy csak az egyedi (ismétlődés nélküli) találatokat kapjuk meg.

8. **Szűrő adatok előkészítése:** Külön lekérdezésekkel begyűjtjük a szűrőkhöz szükséges egyedi adatokat, például a márkkák, helyszínek, évjáratok, vagy sebességváltó típusok listáját.
9. **Adatok átadása a nézetnek:** Az összes összegyűjtött adatot (motorok és szűrők) elküldjük a nézetnek ([view](#)), hogy a felhasználó számára megjeleníthetők legyenek.

Component	SQL Equivalent	Description
Base Query	<code>SELECT motorcycles.*, loans.rentalDate, loans.returnDate</code>	Fetches motorcycles not marked as <code>isInService</code> .
Left Join with Loans	<code>LEFT JOIN loans ON motorcycles.id = loans.motorcycles_id AND loans.id IN (...)</code>	Joins with the latest loan record for each motorcycle using a subquery.
Date Range Filter	<code>WHERE (loans.rentalDate IS NULL OR loans.returnDate < ? OR loans.rentalDate > ?)</code>	Excludes motorcycles rented during the specified dateStart-dateEnd range.
Distinct Results	<code>SELECT DISTINCT motorcycles.*</code>	Ensures no duplicate motorcycle records.

A részletek gombra kattintva az oldal átdob minket egy másik oldalra, ahol az adott motornak az adatait tekinthetjük meg részletesen.

Ennek az elérési útvonala:

```
Route::get(uri: "/motors/{motor}", action: [MotorcycleController::class, "show"])->name(name: "motors.show");
```

Innen től csak a motor id-jával dolgozunk, és annak adatait jelenítjük meg.

```
$motor = Motorcycle::findOrFail(id: $id);  
return view(view: 'motors.show', data: compact(var_name: 'motor'));
```

Amint a felhasználó meggyőződött arról, hogy ezt a motort szeretné, rányom a bérlet gombra, ami ezt az elérési útvonalat tartalmazza:

```
Route::get(uri: 'motors/{motor}/tools', action: [ToolController::class, 'index'])->name(name: 'tools.index');
```

Itt a bérletshez adja hozzá a felhasználó minden eszközöt szeretne a motorja mellé, és a kiválasztott dátum intervallumot, hogy mely időszakban szeretné a motort bérelni.

Itt a rendszer az alábbiakban működik:

1. Motor adatainak lekérése: A megadott motor azonosító (`id`) alapján lekérjük a motor adatait az adatbázisból. Ha nem található ilyen azonosító, hibát dob.
2. Eszközök listázása: Lekérjük az összes elérhető eszközt (`tools`) az adatbázisból. Ez általában a motorhoz kapcsolódó kiegészítők vagy felszerelések listája.

3. Foglalt dátumok lekérése: Az adott motorhoz tartozó kölcsönzési időszakokat (`rentalDate` és `returnDate`) gyűjtjük össze, majd:

- **Időszakok felosztása:** Az egyes kölcsönzések kezdő- és záró dátumai között lévő összes napot meghatározzuk.
- **Formázás:** Az időpontokat `Y-m-d` formátumra alakítjuk, hogy egyszerű JSON-ként továbbadhatók legyenek.
- **Duplikációk eltávolítása:** Eltávolítjuk az ismétlődő dátumokat, hogy minden foglalt nap csak egyszer szerepeljen.
- **Értékek kigyújtése:** A rendezett, ismétlődés nélküli foglalt dátumok listáját alakítjuk ki.

4. Adatok átadása a nézetnek: A nézetnek (`view`) továbbítjuk az összegyűjtött adatokat:

- Az összes eszközt (`tools`),
- Az adott motor adatait (`motor`),

A foglalt dátumokat JSON-kompatibilis formában (`bookedDates`).

```
//Motor adatainak lekérése az id alapján
$motorData = Motorcycle::findOrFail(id: $motor);

//Eszközök lekérése
$tools = Tool::all();

//Foglalt dátumok lekérése csak az adott motorhoz
$bookedDates = Loan::where(column: 'motorcycles_id', operator: $motor) //Szűrés motor id alapján
    ->get(columns: ['rentalDate', 'returnDate'])
    ->flatMap(callback: function (Loan $loan): Collection {
        return collect(value: Carbon::parse(time: $loan->rentalDate)->daysUntil(endDate: $loan->returnDate))
            ->map(callback: fn(TValue $date): mixed => $date->format('Y-m-d'));
    })
    ->unique()
    ->values();

//Motor adatainak és eszközök átadása a nézetnek
return view(view: 'pages.tools', data: [
    'tools' => $tools,
    'motor' => $motorData,
    'bookedDates' => $bookedDates->toArray(), //JSON kompatibilis tömb átadása
]);
```

Variable	Type	Example Value	Use Case
\$tools	Collection	[{"id":1,"name":"Helmet"},...]	Display available tools for selection.
\$motor	Motorcycle	{"id":5,"brand":"Yamaha",...}	Show motorcycle details on the page.
\$bookedDates	Array	["2025-05-01","2025-05-02",...]	Disable booked dates in a calendar UI.

Így a következő oldalon, amelyet a tovább gomb megnyomásával érünk el, tudjuk alkalmazni a megadott adatokat. Eszközök, azok méretei és a kibérlési intervallum.

Amelynek az elérési útvonala:

```
Route::get(uri: 'motors/{motor}/tools/summary_page', action: [MotorcycleController::class, 'showData'])->name(name: 'pages.summary_page');
```

Ezen az oldalon összesítjük a bérleshez tartozó adatokat.

1. A motor adatait
2. felhasználó adatait
3. A kapcsolt eszközök adatait

Mindemellett bekalkulálva a bérlesnek a végösszegét levonva a kedvezményt, ha van.

Az árba beletartozik az eszközök utáni kaució és az eszközök napi bérlesi ára is.

Innen a rendelés megerősítése gombra nyomva a rendelés rögzítésre kerül.

Ennek az elérési útvonala:

```
Route::post(uri: 'motors/{motor}/tools/summary_page/final_page', action: [MotorcycleController::class, 'store'])->name(name: 'pages.final_page');
```

A rendelés mentésének több táblába való művelete:

1. Felhasználói azonosító lekérése: Az aktuális bejelentkezett felhasználó azonosítóját (`userId`) lekérjük az `Auth::id()` függvénnyel.
2. **Session adatok lekérése:** A motorbérleshez szükséges adatokat, mint a bérles kezdő- és végdátuma (`startDate`, `endDate`), illetve a különböző eszközök darabszámát és méretét (`sisakdb`, `ruhadb`, `cipodb`, `sisakmeret`, `ruhameret`, `cipomeret`), a session-ből töltjük be.

```
$startDate = session(key: 'startDate');
$endDate = session(key: 'endDate');

$sisakdb = session(key: 'sisakdb');
$ruhadb = session(key: 'ruhadb');
$cipodb = session(key: 'cipodb');

$sisakmeret = session(key: 'sisakmeret');
$ruhameret = session(key: 'ruhameret');
$cipomeret = session(key: 'cipomeret');
```

3. **Rendelési azonosító generálása:** Egyedi rendelési azonosítót (`orderId`) hozunk létre, amely az `ORD-` előtagból és egy véletlenszerű, 8 karakter hosszú nagybetűs karakterláncból áll.

```
$orderId = 'ORD-' . Str::upper(value: Str::random(length: 8));
```

4. **Motorbérles mentése:** Létrehozunk egy új `MotorRental` példányt, amely tartalmazza a bérleshez kapcsolódó adatokat, például:

- Felhasználói azonosító (`users_id`),

- Motor azonosító (`motorcycles_id`),
- Bérlés kezdő- és végdátuma,
- Egyedi rendelési azonosító. Továbbá, az `gaveDown` mezőt alapértelmezetten 0-ra állítjuk, jelezve, hogy az eszközöket még nem adták le.

```
$motorRental = new MotorRental();
$motorRental->users_id = $userId;
$motorRental->motorcycles_id = $motorId;
$motorRental->rentalDate = $startDate;
$motorRental->returnDate = $endDate;
$motorRental->orders_id = $orderId;

$motorRental->gaveDown = 0;

$motorRental->save();
```

5. Eszközök ellenőrzése: Ha bármelyik eszközből van megadva darabszám (sisak, ruha, cipő), akkor az adatbázisból lekérjük az összes elérhető eszközt.

6. Eszközök szűrése és hozzárendelése:

- Az eszközöket (`tools`) típus és méret alapján ellenőrizzük.
- Megszámoljuk, hogy az adott típusból és méretből mennyi szükséges.
- Az eszköz azonosítókat (`toolId`) hozzáadjuk a megfelelő listához.

7. Kölcsönzés azonosítójának lekérése: Lekérjük a kölcsönzések közül a legnagyobb azonosítót (`maxLoanId`), hogy az új eszközhozzárendelések ehhez a kölcsönzéshez kapcsolódjanak.

```
$matchingToolIds = [];
//Ellenőrizzük, hogy legalább egy eszköz meg van-e adva
if ($ruhadb >= 1 || $sisakdb >= 1 || $cipodb >= 1) {
    //Lekérjük az összes eszközt az adatbázisból
    $tools = DB::table(table: 'tools')->get();

    foreach ($tools as $tool) {
        //Sisak ellenőrzés
        if ($tool->toolName === 'Sisak' && in_array(needle: $tool->size, haystack: $sisakmeret)) {
            $count = array_count_values(array: $sisakmeret][$tool->size] ?? 0; //Számoljuk hányszor kell ez a méret
            for ($i = 0; $i < $count; $i++) {
                $matchingToolIds[] = $tool->id;
            }
        }

        //Ruházat ellenőrzés
        if ($tool->toolName === 'Protektoros Ruha' && in_array(needle: $tool->size, haystack: $ruhameret)) {
            $count = array_count_values(array: $ruhameret][$tool->size] ?? 0; //Számoljuk hányszor kell ez a méret
            for ($i = 0; $i < $count; $i++) {
                $matchingToolIds[] = $tool->id;
            }
        }

        //Cipő ellenőrzés
        if ($tool->toolName === 'Cipő' && in_array(needle: $tool->size, haystack: $cipomeret)) {
            $count = array_count_values(array: $cipomeret][$tool->size] ?? 0; //Számoljuk hányszor kell ez a méret
            for ($i = 0; $i < $count; $i++) {
                $matchingToolIds[] = $tool->id;
            }
        }
    }

    //Lekérjük a legnagyobb kölcsönzés ID-t
    $maxLoanId = DB::table(table: 'loans')->max(column: 'id');

    //A megfelelő eszközök hozzárendelése a kölcsönzéshez
    foreach ($matchingToolIds as $toolId) {
        DeviceSwitch::create(attributes: [
            'loans_id' => $maxLoanId,
            'tools_id' => $toolId,
        ]);
    }
}
```

8. Eszközök mentése: A szűrés után az összes megfelelő eszközt a kölcsönzéshez rendeljük a **DeviceSwitch** modellel, ami az eszközök és a kölcsönzések közötti kapcsolatot kezeli.

Parameter/Data	Example Value	Source	Description
motorId	5	URL parameter	ID of the rented motorcycle.
userId	12	Auth	Authenticated user's ID.
startDate/endDate	"2025-05-01"/"2025-05-05"	Session	Rental period dates.
helmetCount/gearCount/shoeCount	2/1/1	Session	Number of selected tools per type (helmet, protective gear, shoes).
helmetSizes/gearSizes/shoeSizes	["M", "L"]	Session	Selected sizes for each tool type.

9. Adatok átadása a nézetnek: Az összegyűjtött adatokat továbbítjuk a végső nézet ([final_page](#)) számára, beleértve:

- A rendelési azonosítót,
- A motor azonosítót,
- A bérlet kezdő- és végdátumát,
- A kiválasztott eszközök azonosítóit,
- És az eszközökhöz kapcsolódó darabszámot és méreteket.

Innen től már csak átvennie kell a felhasználónak a motort az adott időpontban(napon).

The screenshot shows the application's main interface. At the top, there is a navigation bar with links for 'Főoldal', 'Motorok', 'Helyszínek', 'Rólunk', and a user profile for 'Szilágy Falica Katalin'. Below the navigation bar, there are two main sections. On the left, under 'Felhasználói Adattal', it shows the user's personal information: Name (Falica Katalin), Identification Number (Azonosító: 2), Email (Email cím: kato99@gmail.com), and Phone Number (Telefonszám: +06201548762). It also displays a photo of the user and a driving license thumbnail. On the right, under 'Rendelési azonosító: ORD-KOPYGXH2', it shows the rental details: Motor (Honda 400X), Tools (Sisák (S) - Kapcsolva: 2025.04.28 13:30), and a button labeled 'Új eszköz hozzáadása' (Add new tool). The date range for the rental is listed as 2025-05-12 - 2025-05-15.

Ezután a felhasználó a profil menüben megtudja nézni a saját adatait és rendeléseit, amelyeket tud módosítani, utóbbi még törlni is.

Baloldalt az adatok módosítása gombra nyomva egy felugró ablak jelenik meg ahol a felhasználó a meglévő adatait tudja módosítani.

Onnan a mentés gombra nyomva egy functiont hívunk meg melynek alábbi az elérési útvonala:

```
Route::post(uri: '/update-user', action: [UserController::class, 'update'])->name(name: 'updateUserData');
```

Ebben a funkcionben az adatbázisban a felhasználó adatait frissítjük.

- **Aktuális felhasználó lekérése:** Az `Auth::user()` segítségével lekérjük az aktuálisan bejelentkezett felhasználó adatait.

```
$user = Auth::user();
```

1.2.6.6 Adatok validálása

- A felhasználói adatokat a `validate` metódussal ellenőrizzük:
 - `name`: Kötelező, szöveg típusú, maximum 255 karakter.
 - `email`: Kötelező, érvényes e-mail cím, maximum 255 karakter, és egyedi az `users` táblán belül, kivéve a jelenlegi felhasználó e-mail címét.
 - `phoneNumber`: Kötelező, szöveg típusú, maximum 20 karakter.
 - `drivingLicenceImage`, `drivingLicenceImageBack`: Nem kötelező, de ha van, akkor kép típusúnak kell lennie a megadott fájltípusokkal és maximum 2048 KB mérettel.

```
//Validálás
$request->validate(rules: [
    'name' => 'required|string|max:255',
    'email' => 'required|email|max:255|unique:users,email,' . $user->id,
    'phoneNumber' => 'required|string|max:20',
    'drivingLicenceImage' => 'nullable|image|mimes:jpeg,png,jpg,gif|max:2048',
    'drivingLicenceImageBack' => 'nullable|image|mimes:jpeg,png,jpg,gif|max:2048',
]);
```

1.2.6.7 Felhasználói adatok frissítése

- Az `Auth` által lekért felhasználó adatait (`name`, `email`, `phoneNumber`) frissítjük a kérésből (`$request`) érkező adatokkal.

```
//Felhasználói adatok frissítése
$user->name = $request->name;
$user->email = $request->email;
$user->phoneNumber = $request->phoneNumber;

$updated = false;
```

1.2.6.8 Képek kezelése

- Ellenőrizzük, hogy a kérés tartalmazza-e az adott fájlt (`hasFile`).
- Ha igen, akkor a fájlt az `store` metódussal feltöltsük a `public` tárhelyre, a `driving_licence_images` mappába.
- Az új fájl elérési útját eltároljuk a felhasználó megfelelő attribútumában (`drivingLicenceImage` vagy `drivingLicenceImageBack`).

Az `updated` változót igazra állítjuk, jelezve, hogy történt változtatás.

```
//Driving licence image feltöltés és törlés
if ($request->hasFile(key: 'drivingLicenceImage')) {

    //Új fájl mentése
    $filePath = $request->file(key: 'drivingLicenceImage')->store(path: 'driving_licence_images', options: 'public');
    $user->drivingLicenceImage = $filePath;
    $updated = true;
}

//Driving licence image back feltöltés és törlés
if ($request->hasFile(key: 'drivingLicenceImageBack')) {

    //Új fájl mentése
    $filePathBack = $request->file(key: 'drivingLicenceImageBack')->store(path: 'driving_licence_images', options: 'public');
    $user->drivingLicenceImageBack = $filePathBack;
    $updated = true;
}
```

1.2.6.9 Jogosítvány státusz frissítése

Ha történt bármilyen frissítés (például új kép feltöltése), akkor a `drivingLicenceReal` értékét 0-ra állítjuk, jelezve, hogy az új jogosítvány képeket még nem ellenőrizték.

```
//Ha történt módosítás (kép frissítés), akkor állítsuk a drivingLicenceReal értékét 0-ra
if ($updated) {
    $user->drivingLicenceReal = 0;
}
```

1.2.6.10 Adatok mentése

A felhasználó adatait az adatbázisba mentjük a `save` metódussal.

Visszairányítás: Sikeres frissítés esetén a felhasználót visszairányítjuk az előző oldalra, és egy success üzenetet küldünk, amely tájékoztatja, hogy az adatok sikeresen frissültek.

```
//Mentés
$user->save();

return redirect()->back()->with(key: 'success', value: 'Adatok frissítve');
```

Component	Action
Basic Info	Updates name, email, and phoneNumber from request.
Image Uploads	- Stores new images to public/driving_licence_images. - Updates DB paths.
Verification Reset	Sets drivingLicenceReal=0 if license images are updated.

1.2.6.11 Rendelési előzmények

A felhasználó megtudja tekinteni a korábbi rendeléseit.

Ehhez meghívtunk egy másik funkciót melynek elérési útvonala:

```
Route::get(uri: '/userProfile', action: [UserController::class, 'showLoans'])->name(name: 'userProfile');
```

Itt lekérjük azokat az adatokat a loans táblából, ahol a felhasználó id-ja megegyezik az users_id-val. Hozzáadva azokat az eszközöket (ha vannak) amiknek a rendelési id-ja megegyezik a felhasználó foglalásának id-jával.

```
$userId = auth()->id();

$loans = Loan::with(relations: [
    'deviceSwitches.tool',
    'user',
    'motorcycle'
])->where(column: 'users_id', operator: $userId)
->get()
->map(callback: function (Loan $loan): array {
    return [
        'user_name' => $loan->user->name,
        'orders_id' => $loan->orders_id,
        'motorcycle' => [
            'brand' => $loan->motorcycle->brand,
            'type' => $loan->motorcycle->type,
        ],
        'tools' => $loan->deviceSwitches->map(function ($switch): array {
            return [
                'tool_id' => $switch->tool->id,
                'tool_name' => $switch->tool->toolName,
                'tool_size' => $switch->tool->size,
                'connected_at' => $switch->created_at->format('Y.m.d H:i'),
            ];
        }),
        'rental_period' => [
            'rentalDate' => $loan->rentalDate,
            'returnDate' => $loan->returnDate,
        ],
    ];
});

//Lekérjük az összes elérhető eszközt
$availableTools = Tool::all();

//Méret csoportosítása eszközönként
$availableSizesByTool = Tool::all()->groupBy(groupBy: 'tool_name')->map(callback: function (Collection<int, Tool> $tools): ... {
    return $tools->pluck(value: 'size', key: 'size');
});

//Visszaküldjük a nézetbe az eszközöket és méreteket
return view(view: 'userProfile', data: compact(var_name: 'loans', var_names: 'availableTools', 'availableSizesByTool'));
```

1. **Felhasználó ID lekérése:** Az `auth()->id()` segítségével lekérjük az aktuális felhasználó azonosítóját.
2. **Kölcsönzések lekérése:**
 - Az `Loan` modellből lekérjük a felhasználóhoz kapcsolódó összes kölcsönzést, beleértve a kapcsolódó adatokat:
 - `deviceSwitches` és azok `tool` kapcsolatai.
 - `user` és `motorcycle` adatok.
 - A lekérdezés eredményét átalakítjuk, hogy csak a szükséges adatokat tartalmazza:
 - Felhasználó neve, rendelés azonosítója, motor adatai (márka, típus), eszközök adatai (név, méret, csatlakozás dátuma), és a kölcsönzési időszak.
3. **Elérhető eszközök lekérése:** Lekérjük az összes eszközt a `Tool` modellből.
4. **Eszközök méreteinek csoportosítása:** Az eszközöket név szerint csoportosítjuk, és a méreteket kilistázzuk.

5. **Nézet visszatérése:** A `userProfile` nézetnek átadjuk a kölcsönzéseket, az elérhető eszközöket, és azok csoportosított méreteit (`loans`, `availableTools`, `availableSizesByTool`).

Field	Source	Format/Example
<code>user_name</code>	<code>\$loan->user->name</code>	"John Doe"
<code>orders_id</code>	<code>\$loan->orders_id</code>	"ORD-ABC123"
<code>motorcycle</code>	<code>\$loan->motorcycle</code>	{ brand: "Yamaha", type: "MT-07" }
<code>tools</code>	<code>\$loan->deviceSwitches</code>	Array of { tool_name: "Helmet", size: "M" }
<code>rental_period</code>	<code>rentalDate/returnDate</code>	{ rentalDate: "2025-05-01", returnDate: "2025-05-05" }
<code>created_at</code>	Loan creation timestamp	"2025-04-30 14:30:00"

Megjelenítés mellett tudjuk törölni a foglalást, módosítani és törölni az eszközöket. Ezekre külön-külön function-t alkalmazunk, amelyeknek az elérési útvonala:

```
Route::delete('userProfile/delete-order/{orderId}', action: [UserController::class, 'deleteOrder'])->name(name: 'deleteOrder');

Route::delete('user-profile/tools/{tool}', action: [UserController::class, 'destroy'])->name(name: 'deleteTool');

Route::post('user-profile/{orderId}/add-tool', action: [UserController::class, 'addToolToOrder'])->name(name: 'addToolToOrder');
```

1.2.6.12 Eszköz törlése

```
//Megkeressük az első device_switch rekordot, amelyhez az adott eszköz tartozik
$deviceSwitch = DeviceSwitch::where(column: 'tools_id', operator: $tool->id)->first();

if ($deviceSwitch) {
    //Az első találat törlése
    $deviceSwitch->delete();

    return redirect()->route(route: 'userProfile')->with(key: 'success', value: 'Eszköz törlésre került.');
} else {
    return redirect()->route(route: 'userProfile')->with(key: 'error', value: 'Eszköz nem található a rendelésben.');
}
```

- Eszköz keresése:** A `DeviceSwitch` modellből megkeressük az első rekordot, amely az adott eszközhöz tartozik (`tools_id` alapján).
- Eszköz törlése:**
 - Ha találunk egy megfelelő rekordot, azt töröljük az adatbázisból.
 - Ezután visszairányítjuk a felhasználót a `userProfile` oldalra egy sikerüzenettel.
- Hiba kezelése:**
 - Ha nem találunk megfelelő rekordot, akkor egy hibauzenettel térünk vissza a `userProfile` oldalra.

Component	Description
\$tool	Injected Tool model instance (target tool to remove).
DeviceSwitch::where()	Finds the first device_switches record linked to the tool.
delete()	Removes the pivot record (does not delete the tool itself).

1.2.6.13 Eszköz hozzáadása

- Rendelés keresése:** A Loan modellből lekérjük az első rekordot az `orders_id` alapján.

```
//Megkeressük a rendelést az orders_id alapján
$order = Loan::where(column: 'orders_id', operator: $orderId)->first();
```

- Eszköz ellenőrzése:**

- Megkeressük az eszközt az adatbázisban az `tool_id` alapján.
- Ha az eszköz létezik, az ID-ja alapján meghatározzuk, hogy melyik eszköz típusba tartozik (sisak, protektoros ruha, cipő).

```
//Megkeressük az eszközt az ID alapján
$tool = Tool::find(id: $request->input(key: 'tool_id'));

if ($tool) {
    //Eszköztípusok ID alapú csoportosítása
    $ranges = [
        'sisak' => [1, 5],
        'protektoros ruha' => [6, 10],
        'cipő' => [11, 18]
    ];

    //Ellenőrizzük, hogy az eszköz id-je melyik típusba tartozik
    $tulajdonosTípus = null;
    foreach ($ranges as $type => $range) {
        if ($tool->id >= $range[0] && $tool->id <= $range[1]) {
            $tulajdonosTípus = $type;
            break;
        }
    }
}
```

- Eszköz darabszámának ellenőrzése:**

- Ellenőrizzük, hogy az adott típusú eszközből hány darab kapcsolódik már a rendeléshez.

- Ha már elérte a megengedett maximumot (2 db), akkor hibát jelzünk.

```

if ($tulajdonosTípus) {
    //Megszámoljuk, hány eszköz van már a rendeléshez kapcsolva az adott típusból
    $jelenlegiDarab = $order->deviceSwitches()
        ->whereHas(relation: 'tool', callback: function ($query) use ($tulajdonosTípus, $ranges): void {
            $range = $ranges[$tulajdonosTípus];
            $query->where('id', '>=', $range[0]) // Minimum ID a típushoz
                ->where('id', '<=', $range[1]); // Maximum ID a típushoz
        })
        ->count();

    //Ha már elérte a maximumot (2 db), hibaüzenetet adunk vissza
    if ($jelenlegiDarab >= 2) {
        return redirect()->back()->with(
            key: 'error',
            value: "Maximum 2 db {$tulajdonosTípus} eszköz bérélhető a rendeléshez!"
        );
    }
}

```

4. Eszköz hozzáadása:

- Ha még nincs elérve a limit, az új eszközt hozzáadjuk a rendeléshez a `deviceSwitches` táblában.

```

//Ha még nem érte el a maximumot, hozzáadjuk az eszközt
$order->deviceSwitches()->create(attributes: [
    'tools_id' => $tool->id,
    'loans_id' => $order->id,
    'created_at' => now(),
]);

```

Parameter	Type	Example	Source	Description
ordersId	Integer	1	URL parameter	Unique order identifier.
tool_id	Integer	3	Request input	ID of the tool to add.

```

//Ha még nem érte el a maximumot, hozzáadjuk az eszközt
$order->deviceSwitches()->create(attributes: [
    'tools_id' => $tool->id,
    'loans_id' => $order->id,
    'created_at' => now(),
]);

```

1.2.6.14 Rendelés törlése

1. **Rendelés keresése:** Az `orders_id` alapján megkeressük az adott rendelést a `Loan` modellből.

```

//A rendelést az orders_id alapján keresjük
$order = Loan::where(column: 'orders_id', operator: $ordersId)->first();

```

2. Kapcsolódó eszközök törlése:

- A rendeléshez kapcsolódó összes eszközt (`deviceSwitches`) töröljük.

```
$order->deviceSwitches()->delete();
```

3. Rendelés törlése:

- A rendelést (`order`) is töröljük az adatbázisból.

```
$order->delete();
```

Operation	SQL Equivalent	Effect
<code>deviceSwitches()->delete()</code>	<code>DELETE FROM device_switches WHERE loans_id=?</code>	Deletes all linked tools.
<code>order->delete()</code>	<code>DELETE FROM loans WHERE id=?</code>	Permanently deletes the loan order.

1.3 A mobilapplikáció bemutatása

1.3.1 Működése

A laravelben megírt funkciókat kértük le API segítségével. Ezeket kezdtük el alkalmazni a telefonos alkalmazásunkban. Így nem kellett újra az adatbázist feldolgozni a rendszerünkben, hanem "újra használjuk" a korábban használt műveleteket.

Elsősorban feldolgozzuk, hogy milyen adatokat szeretnénk tárolni a fetchelés során, például a felhasználói adatokat:

```
interface IUser {
    id: number;
    name: string;
    email: string;
    phoneNumber: string;
    drivingLicenceNumber: string;
    drivingLicenceType: string;
    drivingLicenceImage?: string;
    drivingLicenceImageBack?: string;
}
```

Itt megadjuk a rendszernek, hogy a felhasználó nevét, email címét és egyéb adatait szeretnénk később alkalmazni.

Ezt következően készül el egy "function", amelyben lekérjük api-n keresztül a kívánt adatokat.

1.3.2 Bejelentkezés

Paraméterek

- `email(string)`: A felhasználó email címe.
- `password(string)`: A felhasználó jelszava.

Működés

Kezdő állapotok beállítása: A `loading` állapot `true-ra` állítása és a `loginError` törlése, ha van.

```
async (email: string, password: string) => {
  try {
    setLoading(true);
    setLoginError("");
  }
```

1. **Kérés küldése:** A `POST` kérést küldi az API-nak (`http://127.0.0.1:8000/api/login`), amely tartalmazza a felhasználó email címét és jelszavát JSON formátumban.

```
const response = await fetch("http://127.0.0.1:8000/api/login", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  body: JSON.stringify({ email, password }),
});

const data = await response.json();
```

2. **Válasz feldolgozása:**

- **Sikeres válasz:** Ha a válasz sikeres, akkor a választ JSON formátumban dolgozza fel, és a `token` és `userId` értékeit elmenti a `localStorage`-ba. Ezt követően lekéri a felhasználói adatokat és átirányítja a felhasználót a dashboard oldalra.
- **Sikertelen válasz:** Ha a válasz nem sikeres, a hibaüzenetet jeleníti meg a felhasználónak.

```

    if (response.ok) {
      localStorage.setItem("authToken", data.token);
      localStorage.setItem("userId", data.userId);
      await fetchUserData();
      navigate("/dashboard");
    } else {
      setLoginError(data.message || "Bejelentkezés sikertelen.");
    }
  } catch (error) {
    setLoginError("Hiba a bejelentkezés során");
    console.error("Login error:", error);
  } finally {
    setLoading(false);
  }
},
[navigate, fetchUserData]

```

Ezután a felhasználó megtudja tekinteni az adatait, a korábbi rendeléseit, és azokat törölheti, ha még a megadott időintervallumon belül van.

Ha bejelentkezett a felhasználó a bejelentkezés gomb helyére egy kijelentkezés gomb kerül, amelyre nyomva kilépteti a felhasználót.

```

const handleLogout = useCallback(() => {
  localStorage.removeItem("authToken");
  localStorage.removeItem("userId");
  setUserData(null);
  setLoans([]);
  setTools([]);
  navigate("/login");
}, [navigate]);

```

1.3.3 Kijelentkezés

Működés

- Adatok eltávolítása a `localStorage`-ból:** A függvény eltávolítja a `authToken` és `userId` kulcsokat a `localStorage`-ból, hogy megszüntesse az autentikációs adatokat.
- Felhasználói adatok törlése:** A `setUserData(null)` hívásával törli a felhasználói adatokat.
- Kapcsolódó listák törlése:** A `setLoans([])` és `setTools([])` hívásokkal törli a hitelekkel és eszközökkel kapcsolatos adatokat.
- Navigáció:** A felhasználót átirányítja a `/login` oldalra a `navigate("/login")` segítségével.

Korábban említve, a felhasználó bejelentkezést követően megtudja tekinteni az adatait, amelyet api-nak a segítségével kérünk le.

A bejelentkezéskor beléptettük a felhasználót és annak a profinak az id-ját alkalmazzuk a következő lekéréskor.

1.3.4 Profilbetöltés

Működés:

1. **Autentikációs adatok ellenőrzése:** A függvény lekéri a `localStorage`-ból az `authToken` és `userId` adatokat. Ha bármelyik hiányzik, átirányítja a felhasználót a bejelentkezési oldalra, ha a profil fülre kattint.

```
const token = localStorage.getItem("authToken");
const userId = localStorage.getItem("userId");
if (!token || !userId) {
  navigate("/login");
  return;
}
```

2. **API kérés küldése:** Ha a token és a felhasználói ID rendelkezésre áll, a függvény egy GET kérést küld az API-nak (`http://localhost:8000/api/userProfile/${userId}`), az autentikációs tokent az `Authorization` fejlécben küldve.

```
try {
  setLoading(true);
  const response = await fetch(
    `http://localhost:8000/api/userProfile/${userId}`,
    {
      method: "GET",
      headers: {
        Authorization: `Bearer ${token}`,
        "Content-Type": "application/json",
      },
    }
  );
  const data = await response.json();
}
```

3. **Válasz feldolgozása:**

Sikeres válasz: Ha a válasz sikeres, a választ JSON formátumban dolgozza fel, és frissíti az alkalmazás állapotát a `setUserData`, `setLoans` és `setTools` hívásokkal. A hibát (`error`) nullázzák.

```
if (response.ok) {
  setUserData(data.user || null);
  setLoans(data.loans || []);
  setTools(data.availableTools || []);
  setError("");
}
```

Sikertelen válasz: Ha a válasz nem sikeres, a hibaüzenetet a `setError` hívással tárolja.

```

    } else {
      setError(data.msg || "Hibás lekérés");
    }
  } catch (error) {
    setError("Hibás lekérés");
    console.error("Fetch error:", error);
  } finally {
    setLoading(false);
  }
}

```

Itt még egyelőre a felhasználó semmi mászt nem tud csinálni, csak a rendelés törlését elvégezni. Viszont további lehetőségekre is hagytunk helyet, megírva már a többi beépíthető function API kérését.

Ezen kívül a felhasználó meg tud tekinteni egy katalógust, amelyben a rendszerünkben lévő motorokat tudja megttekinteni.

The screenshot shows a user interface for renting motorcycles. At the top, there's a navigation bar with links for 'Profil', 'Főoldal', 'Foglalj most!', and 'Kijelentkezés'. Below the navigation, the heading 'Elérhető motorok' is centered above a grid of twelve motorcycle images. Each motorcycle is displayed in a separate card with its name, year, transmission type, location, and a short description.

Model	Year	Transmission	Location	Description
Aprilia - Mana 850	2012	Automata	1096 Budapest, Haller utca 89.	
Aprilia - Tuono 125	2017	Manuális (6 fokozatú)	1195 Budapest, Ady Endre út 134.	
Aprilia - SR 160	2021	Automata	1103 Budapest, Győmrői út 50.	
Aprilia - Caponord 1200	2015	Manuális (6 fokozatú)	1044 Budapest, Megyeri út 20.	
Aprilia - RS 125	2010	Manuális (6 fokozatú)	1126 Budapest, Böszörményi út 38.	
Aprilia - RSV4 Factory	2010	Manuális (6 fokozatú)	1126 Budapest, Böszörményi út 38.	
Aprilia - Tuono 660	2023	Manuális (6 fokozatú)	1161 Budapest, Rákosi út 88.	
BMW - C 400 GT	2020	Automata	1096 Budapest, Haller utca 69.	
BMW - C Evolution	2020	Automata	1211 Budapest, Szallító utca 6.	
BMW - Concept 101	2022	Manuális (6 fokozatú)	1064 Budapest, Podmaniczky utca 63.	
BMW - F 800 ST	2010	Manuális (6 fokozatú)	1148 Budapest, Rógarasi út 45.	
BMW - G 310 R	2019	Manuális (6 fokozatú)	1151 Budapest, Karolyi Sándor ut 76.	

Ennek a lekérése az alábbként működik:

- URL és szűrők generálása:** A függvény létrehoz egy URL-t a motorok lekéréséhez (<http://localhost:8000/api/motorcycles>), és hozzáadja a szűrőket a URL keresési paramétereihez, ha azok rendelkezésre állnak.
- API kérés küldése:** A függvény egy **GET** kérést küld az API-nak a generált URL-lel. Az autentikációs tokent az Authorization fejlécben küldi el.

```

try {
  setLoading(true);
  const url = new URL("http://localhost:8000/api/motorcycles");

  if (filters.brand) url.searchParams.append("brand", filters.brand);
  if (filters.year) url.searchParams.append("year", filters.year);
  if (filters.gearbox) url.searchParams.append("gearbox", filters.gearbox);
  if (filters.fuel) url.searchParams.append("fuel", filters.fuel);
  if (filters.location) url.searchParams.append("location", filters.location);
  if (filters.dateStart) url.searchParams.append("dateStart", filters.dateStart);
  if (filters.dateEnd) url.searchParams.append("dateEnd", filters.dateEnd);

  const response = await fetch(url.toString(), {
    method: "GET",
    headers: {
      Authorization: `Bearer ${token}`,
      "Content-Type": "application/json",
    },
  });

  const data = await response.json();
}

```

3. Válasz feldolgozása:

- **Sikeres válasz:** Ha a válasz sikeres, a választ JSON formátumban dolgozza fel, és beállítja a `motorcycles` adatokat, valamint a `filters` szűrőket. A hibát (`error`) nullázza.
- **Sikertelen válasz:** Ha a válasz nem sikeres, a hibaüzenetet a `setError` hívással tárolja.

```

if (response.ok) {
  setMotorcycles(data.motorcycles || []);
  setFilters(data.filters);
  setError("");
} else {
  setError(data.msg || "Sikertelen lekérés");
}
} catch (error) {
  setError("Hiba a lekérés során");
  console.error("Fetch error:", error);
} finally {
  setLoading(false);
}

```

1.4 Adminisztrációs alkalmazás bemutatása

1.4.1 Adminisztrátori alkalmazás

1.4.1.1 Technológiai háttér

Az adminisztrátori alkalmazás a .NET MAUI keretrendszerben készült, C# nyelven, **MVVM (Model-View-ViewModel)** architektúra alkalmazásával. Az adatkezelés RESTful API-kon keresztül történik.

1.4.1.2 Alkalmazásindítás és admin inicializálás

Az alkalmazás elindítása után az alábbi logika fut le:

- Lekérdezés történik az adatbázisban található admin tábla tartalmára.
- Ha nincs admin rögzítve, az alkalmazás automatikusan létrehoz egy **alapértelmezett adminisztrátort** az alábbi adatokkal:
 - Email: teszt@gmail.com
 - Jelszó: Password
- Ez biztosítja, hogy legalább egy felhasználó képes legyen belépni és új adminokat felvenni.

1.4.1.3 Bejelentkezési logika

A bejelentkezés folyamata több ellenőrzési lépésből áll:

1. Az e-mail cím létezik-e az adatbázisban.
2. A megadott jelszó megfelel-e a rögzített jelszónak.
3. Az admin fiók nincs-e deaktiválva.
4. Sikeres hitelesítés esetén a rendszer az admin **beosztása alapján** irányítja át a megfelelő oldalra.

```

//Load Admin
1 reference
private async void LoadAdmin()
{
    Admins = await AdminAPIService.GetLogsDataAsync();
    if (Admins.Count == 0)
    {
        string name = "Teszt Elek";
        string email = "teszt@gmail.com";
        string password = "Password";
        int jobStatus = 0;
        List<LogModel> Logs = new List<LogModel>();

        string APasswordHash = BCrypt.Net.BCrypt.HashPassword(password);

        var parameters = new Dictionary<string, object>
        {
            {"name", name},
            {"email", email},
            {"password", APasswordHash},
            {"jobStatus", jobStatus},
        };

        await AdminService.AddAdminIntoDatabaseAsync(parameters, jobStatus);
        Admins = await AdminAPIService.GetLogsDataAsync();
    }
    AllItems = Admins.Select(admin => admin.email).ToList();
}

```

```

[RelayCommand]
10 references
public async Task LoginAsync()
{
    bool goodPair = false;

    foreach (AdminModel admin in Admins)
    {
        if (admin.email == EmailInput.ToLower()
            && BCrypt.Net.BCrypt.Verify(PasswordInput, admin.password)
            && admin.deleted_at == null)
        {
            LoggedInUser = admin;
            goodPair = true;
            if (admin.jobStatus == 0)
            {
                var parameters = new Dictionary<string, object>
                {
                    { "LoggedInUser", admin }
                };
                await Shell.Current.GoToAsync(nameof>ShowAdminsPage), parameters);
            }
            else if (admin.jobStatus == 1)
            {
                var parameters = new Dictionary<string, object>
                {
                    { "LoggedInUser", LoggedInUser }
                };
                await Shell.Current.GoToAsync(nameof>ShowMotorcyclePage), parameters);
            }
            else if (admin.jobStatus == 2)
            {
                var parameters = new Dictionary<string, object>
                {
                    { "LoggedInUser", LoggedInUser }
                };
                await Shell.Current.GoToAsync(nameof>SwitchPage), parameters);
            }
        }
        else {
        }
    }
    if (goodPair == false)
    {
        await Shell.Current.DisplayAlert("Hiba", "Rossz Jelszó-Email Páros", "OK");
    }
}

```

1.4.1.4 Autocomplete funkció

Az email mező gépelése közben az alkalmazás **automatikusan javaslatokat jelenít meg** a korábban rögzített email címek közül, az aktuálisan beírt karakterek alapján (prefix-alapú keresés).

```
2 references
private void FilterSuggestions()
{
    var filtered = AllItems.Where(item => !string.IsNullOrWhiteSpace(EmailInput)
        && item.StartsWith(EmailInput, StringComparison.OrdinalIgnoreCase)).ToList();

    Suggestions.Clear();
    foreach (var item in filtered)
    {
        Suggestions.Add(item);
    }
}
```

1.4.1.5 Munkamenet-kezelés

- Az alkalmazás **nyilvántartja, ki van bejelentkezve**, és képes naplózni a felhasználó tevékenységeit.
- A felhasználói információkat csak a kijelentkezés után felejti el a rendszer.

```
[RelayCommand]
10 references
public async void NavigationToRentalPages()
{
    var parameters = new Dictionary<string, object>
    {
        { "LoggedInUser", LoggedInUser }
    };
    await Shell.Current.GoToAsync(nameof>ShowUsersPage), parameters);
}
```

1.4.1.6 Módosítások naplózása

Minden adatbázison végrehajtott módosítás rögzítésre kerül egy logs nevű táblában. A naplóbejegyzések a következő adatokat tartalmazzák:

- **Művelet típusa** (pl. új admin létrehozása, törlés, módosítás)
- **Felhasználó azonosítója**, aki a műveletet végezte
- **Időbényeg**, azaz a művelet pontos időpontja

```

string formattedDate = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");
var log = new LogModel
{
    command = $"Sikeresen FRISSITETTE a {motorId}, {type} a motort",
    date = formattedDate,
    admin_id = CurrentAdminId,
};
await adminService.AddLog(log);

```

```

// Add Log to the database
13 references
public async Task AddLog(LogModel log)
{
    try
    {
        var uri = $"http://127.0.0.1:8000/api/AddLog";
        var content = new StringContent(JsonConvert.SerializeObject(log), Encoding.UTF8, "application/json");
        var response = await _httpClient.PostAsync(uri, content);
        if (!response.IsSuccessStatusCode)
        {
            string responseContent = await response.Content.ReadAsStringAsync();
            await Shell.Current.DisplayAlert("Hiba", $"Nem sikerült a hozzáadás. \n Részletek: {responseContent}", "OK");
        }
    }
    catch (Exception ex)
    {
        string filePath = Path.Combine(FileSystem.AppDataDirectory, "error.txt");
        File.WriteAllText(filePath, ex.ToString());
        throw;
    }
}

```

1.4.1.7 Módosítások megerősítése

Minden egyes oldalon, ahol olyan adatok kerülnek megadásra, amelyek az adatbázis módosítását eredményezik, a módosítás elvégzését megelőzően megjelenik egy felugró ablak, ahol az adminnak meg kell erősítenie az akciót.

1.4.1.8 Kijelentkezés

Minden főoldalon található egy Kijelentkezés gomb, amelyet megnyomva elfelejtíti a belépett felhasználót (kitörli a CurrentUser-t), és visszavisz a „ MainPage.xaml”-re, a bejelentkezési felületre.

1.4.2 Szuperadminok oldalai

Ha belépett admin Szuperadmin, akkor a következő műveleteket végezheti el (a következő oldalakhoz lesz hozzáférésre).

1.4.2.1 Szuperadmin főoldal

Ha az adott admin pozíciója Szuperadmin, akkor a rendszer a belépést követően a „ShowAdminsPage.xaml” oldalra viszi, ahol először is a program API (GET metódus) segítségivel lekéri az összes (aktív) admint és az ahhoz tartozó logokat is. Azok az adminok számítanak aktívnak, amelyeknél a „deleted_at” cella értéke nem null érték.

```

// Get all logs from the database
1 reference
public async Task AddAdminIntoDatabaseAsync(Dictionary<string, object> addData, int jobStatus)
{
    try
    {
        var uri = $"http://127.0.0.1:8000/api/admins";

        var content = new StringContent(JsonConvert.SerializeObject(addData), Encoding.UTF8, "application/json");

        var response = await _httpClient.PostAsync(uri, content);

        string jobStatusText = "";

        switch (jobStatus)
        {
            case 0:
                jobStatusText = "SzuperAdmin";
                break;
            case 1:
                jobStatusText = "Szerelő";
                break;
            case 2:
                jobStatusText = "Recepció";
                break;
        }

        if (!response.IsSuccessStatusCode)
        {
            string responseContent = await response.Content.ReadAsStringAsync();

            await Shell.Current.DisplayAlert("Hiba", $"Nem sikerült a hozzáadás. \n Részletek: {responseContent}", "OK");
        }
    }
    catch (Exception ex)
    {
        string filePath = Path.Combine(FileSystem.AppDataDirectory, "error.txt");
        File.WriteAllText(filePath, ex.ToString());
        throw;
    }
}

```

1.4.2.2 Új admin felvétele

Új admin hozzáadása az „AddAdminPage.xaml”-en történik. Az adatok megadását és a HOZZÁADÁS gomb megnyomását követően az alkalmazás ellenőrzést végez. Ellenőrzi., hogy a megadott email cím egyedi-e, hogy email cím-e (tartalmaz-e „@” karaktert), és hogy a beállított jelszóban van-e minimum 1-1 db kis- és nagybetű, szám, illetve speciális karakter és minimum 8 karakter-e a jelszó. Ha minden rendben van, akkor a program Hash-eli a jelszót, majd az API (POST) metódus segítségével hozzáadja az adatbázishoz a létrehozott admint (és természetesen ez az akció is logolásra kerül). Ezt követően visszalép a „ShowAdminsPage.xaml” oldalra.

```

[RelayCommand]
10 references
public async Task AdminAddAsync()
{
    bool x = await Shell.Current.DisplayAlert("Adatellenőrzés", "Minden adatot rendben talált? Kérjük, ellenőrizze még egyszer!");
    if (x)
    {
        if (!string.IsNullOrEmpty(NewName)
            && !string.IsNullOrEmpty(NewEmail)
            && !string.IsNullOrEmpty(NewPassword)
            && NewJobStatus != null)
        {
            bool emailExists = AllAdminData.Any(admin => admin.email.ToLower() == NewEmail.ToLower());

            if (NewEmail.Contains("@") && !emailExists)
            {
                if (NewPassword.Length > 8 && NewPassword.Any(char.IsLower) && NewPassword.Any(char.IsUpper)
                    && NewPassword.Any(char.IsDigit) && NewPassword.Any(c => !char.IsLetterOrDigit(c)))
                {
                    string APasswordHash = BCrypt.Net.BCrypt.HashPassword(NewPassword);

                    var parameters = new Dictionary<string, object>
                    {
                        {"name", NewName},
                        {"email", NewEmail.ToLower()},
                        {"password", APasswordHash},
                        {"jobStatus", NewJobStatus},
                    };
                    await AdminAPIService.AddAdminAsync(parameters, LoggedInUser.Id, LoggedInUser, NewName, NewJobStatus);
                }
                else
                {
                    await Shell.Current.DisplayAlert("Hiba", "A jelszónak legalább 8 karakter hosszúnak kell lennie, és tartalmazni kell mindenféle karaktert.");
                }
            }
            else
            {
                await Shell.Current.DisplayAlert("Hiba", "Az email cím nem érvényes. Hiányzik belőle a '@' karakter vagy más karakterek.");
            }
        }
        else
        {
            await Shell.Current.DisplayAlert("Hiányzó adatok", "Kérjük, töltse ki az összes mezőt: név, email és jelszó.");
        }
    }
}

```

1.4.2.3 Adminok deaktiválása

Korábban felvett adminokat a „ShowAdminsPage.xaml” oldalon lehet deaktiválni. A Deaktiválás gomb megnyomásának a hatására az API (a DELETE metódus segítségével) kitölti a deleted_at mező értékét az aktuális időpontra.

```

1 reference
public async void LoadAdminData()
{
    IsLoading = true;
    await Task.Delay(3000);
    AllAdminData = await AdminAPIService.GetLogsDataAsync();
    if (AllAdminData != null)
    {
        DeletesAdminData = AllAdminData.Where(admin => admin.deleted_at != null).ToList();
        FilteredAdminData = AllAdminData.Where(admin => admin.deleted_at == null).ToList();
        LogFilteredAdminData = AllAdminData.Where(admin => admin.Logs.Any()).ToList();
    }
    IsLoading = false;
}

```

```

//Deactivate Admin from the database
1 reference
public async Task DeleteAdminAsync(int itemId, int currentAdminId, string currentAdminName, AdminModel loggedAdmin)
{
    try
    {
        var uri = $"http://127.0.0.1:8000/api/DeactiveAdmin/{itemId}";

        var response = await _httpClient.DeleteAsync(uri);

        if (response.IsSuccessStatusCode)
        {
            string formattedDate = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");

            var log = new LogModel
            {
                command = $"Sikeresen frissítette az {currentAdminId} Id ",
                date = formattedDate,
                admin_id = currentAdminId,
            };
            await adminService.AddLog(log);

            var parameters = new Dictionary<string, object>
            {
                {"LoggedInUser", loggedAdmin }
            };
            await Shell.Current.GoToAsync(nameof(ShowAdminsPage), parameters);
        }
        else
        {
            string responseContent = await response.Content.ReadAsStringAsync();
            await Shell.Current.DisplayAlert("Hiba", $"Nem sikerült a törlés. \n Részletek: {responseContent}", "OK");
        }
    }
    catch (Exception ex)
    {
        string filePath = Path.Combine(FileSystem.AppDataDirectory, "error.txt");
        File.WriteAllText(filePath, ex.ToString());
        throw;
    }
}

```

1.4.2.4 Deaktivált adminok visszaállítása

A deaktivált adminokat a „DeleteAdminPage.xaml” oldalon lehet megtekinteni. A Visszaállítás gomb megnyomásának a hatására az API (PUT metódus segítségével) végrehajtja a restore parancsot az adott adminon, és a „deleted_at” cella újra null értéket kap.

1.4.2.5 Adminok adatainak módosítása

Adminok adatainak módosítása az „EditAdminPage.xaml” oldalon történik. Itt láthatóak az aktuális adatok (név, email, beosztás), amelyeket a Szuperadmin módosíthat. A jelszó természetesen nem látható (üres mező jelenik meg), ha ezt kitölti, akkor az a jelszó módosítását eredményezi. Ha üresen hagyja a cellát, akkor a jelszó nem módosul. A Módosítás gombra történő kattintást után az alkalmazás ellenőrzést végez, hasonlóan az új admin felvételéhez. Ha minden rendben van, akkor az API (PUT metódus) segítségével módosítja az adatokat az adatbázisban. Amennyiben új jelszó került megadásra, akkor az elsőként Hash-elésre kerül, és úgy lesz elmentve. Ezt követően az alkalmazás visszatér a „ShowAdminsPage.xaml” oldalra.

```

[RelayCommand]
10 references
public async Task AdminUpdate(AdminModel admin)
{
    bool x = await Shell.Current.DisplayAlert("Adatellenőrzés", "Minden adatot rendben talált?", "Igen", "Nem");
    if (x)
    {
        if (!string.IsNullOrEmpty(CurrentUser.name)
            && !string.IsNullOrEmpty(CurrentUser.email)
            && !string.IsNullOrEmpty(CurrentUser.password))
        {
            bool emailExists = allAdminData.Any(admin => admin.email.ToLower() == CurrentUser.email.ToLower()
            && admin.Id != CurrentUser.Id);

            if (CurrentUser.email.Contains("@") && !emailExists)
            {
                if (CurrentUser.password.Length > 8
                    && CurrentUser.password.Any(char.IsLower)
                    && CurrentUser.password.Any(char.IsUpper)
                    && CurrentUser.password.Any(char.IsDigit)
                    && CurrentUser.password.Any(c => !char.IsLetterOrDigit(c)))
                {
                    var updatedata = new Dictionary<string, object>();
                    if (CurrentPassword == "")
                    {
                        CurrentPassword = CurrentUser.password;
                        updatedata = new Dictionary<string, object>
                        {
                            {"name", CurrentUser.name},
                            {"email", CurrentUser.email},
                            {"password", CurrentPassword},
                            {"jobStatus", CurrentUser.jobStatus},
                        };
                    }
                    else
                    {
                        string UPasswordHash = BCrypt.Net.BCrypt.HashPassword(CurrentPassword);
                        updatedata = new Dictionary<string, object>
                        {
                            {"name", CurrentUser.name},
                            {"email", CurrentUser.email},
                            {"password", UPasswordHash},
                            {"jobStatus", CurrentUser.jobStatus},
                        };
                    }
                    await AdminAPIService.UpdateAdminAsync(CurrentUser.Id, updatedata, LoggedInUser.Id, CurrentUser.name, LoggedInUser);
                }
                else
                {
                    await Shell.Current.DisplayAlert("Hiba", "A jelszónak legalább 8 karakter hosszúnak kell lennie, és tartalmaznia kell kisbetűt");
                }
            }
            else
            {
                await Shell.Current.DisplayAlert("Hiba", "Az email cím nem érvényes. Hiányzik belőle a '@' karakter vagy már létezik", "OK");
            }
        }
    }
}

```

```

//Update Admin to the database
1 reference
public async Task UpdateAdminAsync(int itemId, Dictionary<string, object> updateData,
    int currentAdminId, string adminName, AdminModel loggedAdmin)
{
    try
    {
        var uri = $"http://127.0.0.1:8000/api/admins/{itemId}";

        var content = new StringContent(JsonConvert.SerializeObject(updateData), Encoding.UTF8, "application/json");

        var response = await _httpClient.PutAsync(uri, content);

        if (response.IsSuccessStatusCode)
        {
            string formattedDate = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");

            var log = new LogModel
            {
                command = $"Sikeresen MÓDOSÍTOTT A {adminName} admint.",
                date = formattedDate,
                admin_id = currentAdminId,
            };
            await adminService.AddLog(log);

            var parameters = new Dictionary<string, object>
            {
                {"LoggedInUser", loggedAdmin }
            };
            await Shell.Current.GoToAsync(nameof(ShowAdminsPage), parameters);
        }
        else
        {
            string responseContent = await response.Content.ReadAsStringAsync();
            await Shell.Current.DisplayAlert("Hiba", $"Nem sikerült a frissítés. \n Részletek: {responseContent}", "OK");
        }
    }
    catch (Exception ex)
    {
        string filePath = Path.Combine(FileSystem.AppDataDirectory, "error.txt");
        File.WriteAllText(filePath, ex.ToString());
        throw;
    }
}

```

1.4.2.6 Naplóadatok megtekintése

A műveleti naplót a „LogsShowPage.xaml” oldalon lehet megtekinteni, ahol az API (GET metódusával) lekéri az összes admint és logot és kiírja egy felsorolásban.

1.4.3 Szerelők oldalai

Ha belépett admin Szerelő, akkor a következő műveleteket végezheti el (a következő oldalakhoz lesz hozzáférés).

1.4.3.1 Szerelő főoldal

Ha az adott admin pozíciója Szerelő, akkor a rendszer a belépést követően a „ShowMotorcyclePage.xaml” oldalra viszi, ahol először is a program API (GET metódus) segítségivel lekéri és megjeleníti az összes motort, ami a flottában van, kivéve azokat, amelyek szervízben vannak.

```

2 references
private async void LoadMotorAsync()
{
    IsLoading = true;
    await Task.Delay(1000);
    AllMotorcycle = await services.GetAllMotorDataAsync();
    AllInServicesMotorcycle = await services.GetAllInServiceMotorDataAsync();
    IsLoading = false;
}

```

```

internal class AlertExpired : IValueConverter
{
    0 references
    public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
    {
        if (value is DateTime trafficDate && trafficDate < DateTime.Now)
        {
            return MyColor.MyRed;
        }
        return MyColor.MyGreen;
    }

    0 references
    public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)
    {
        throw new NotImplementedException();
    }
}

```

1.4.3.2 Új motor felvétele

Új motor hozzáadása az „AddMotorcyclePage.xaml”-en történik. Az adatok (pl. márka, rendszám, bérlesi ára, motor képe stb.) megadását és a HOZZÁADÁS gomb megnyomását követően az alkalmazás ellenőrzést végez. Ellenőrzi, hogy minden cella ki van-e töltve, valamint elvégzi az egyes adatok validálását (pl. hogy a rendszám formátuma megfelelő-e, hogy a gyártási év reális-e, az árnál és lóerőnél szám szerepel-e stb.). Ha minden rendben van, akkor a program az API (POST) metódus segítségével hozzáadja az adatbázishoz a létrehozott motort. A motor képét feltölti a laravel storage/app/public/img mappába tölti fel. Ezt követően visszalép a „ShowMotorcyclePage.xaml” oldalra.

```

//Add Motor to the database
1 reference
public async Task AddMotorAsync(Dictionary<string, object> addData, string motorBrand,
    string motorType, int CurrentAdminId, AdminModel LoggedAdmin, string imagePath)
{
    try
    {
        var uri = "http://127.0.0.1:8000/api/AddMotor";
        var content = new MultipartFormDataContent();

        // Szöveges mezők hozzáadása
        foreach (var item in addData)
        {
            if (item.Value != null)
            {
                content.Add(new StringContent(item.Value.ToString()), item.Key);
            }
        }

        // Kép feltöltése
        if (!string.IsNullOrEmpty(imagePath) && File.Exists(imagePath))
        {
            var stream = File.OpenRead(imagePath);
            var fileContent = new StreamContent(stream);

            // MIME típus automatikus meghatározása
            var mimeType = GetMimeType(imagePath);
            fileContent.Headers.ContentType = new MediaTypeHeaderValue(mimeType);

            content.Add(fileContent, "image", Path.GetFileName(imagePath));
        }

        var response = await _httpClient.PostAsync(uri, content);

        if (response.IsSuccessStatusCode)
        {
            string formattedDate = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");

            var log = new LogModel
            {
                command = $"Sikeresen HOZZÁADTA a {motorBrand}-{motorType} motort",
                date = formattedDate,
                admin_id = CurrentAdminId,
            };
            await adminService.AddLog(log);

            var parameters = new Dictionary<string, object>
            {
                { "LoggedInUser", LoggedAdmin }
            };
            await Shell.Current.GoToAsync(nameof(ShowMotorcyclePage), parameters);
        }
        else
        {
            string responseContent = await response.Content.ReadAsStringAsync();

            await Shell.Current.DisplayAlert("Hiba", $"Nem sikerült a hozzáadás. Részletek: {responseContent}", "OK");
        }
    }
    catch (Exception ex)
    {
        string filePath = Path.Combine(FileSystem.AppDataDirectory, "error.txt");
        File.WriteAllText(filePath, ex.ToString());
        throw;
    }
}

```

```

// MIME típus meghatározása a fájl kiterjesztése alapján
1 reference
private string GetMimeType(string filePath)
{
    var mimeType = "application/octet-stream";
    var extension = Path.GetExtension(filePath).ToLower();

    switch (extension)
    {
        case ".jpg":
            mimeType = "image/jpg";
            break;
        case ".jpeg":
            mimeType = "image/jpeg";
            break;
        case ".png":
            mimeType = "image/png";
            break;
        case ".gif":
            mimeType = "image/gif";
            break;
        case ".svg":
            mimeType = "image/svg";
            break;
    }

    return mimeType;
}

```

```

[RelayCommand]
10 references
public async Task MotorAddAsync()
{
    bool x = await Shell.Current.DisplayAlert("Adatellenőrzés", "Minden adatot rendben talált?", "Igen", "Nem");
    if (x)
    {
        int year = 0; float Le = 0; float kW = 0; float engine = 0; int place = 0; int price = 0; int deposit = 0; Date
        if (!string.IsNullOrEmpty(NewBrand) && !string.IsNullOrEmpty(NewType) && !string.IsNullOrEmpty(NewLicencePlate)
        && !string.IsNullOrEmpty(NewYear) && !string.IsNullOrEmpty(NewGearbox) && !string.IsNullOrEmpty(NewFuel)
        && !string.IsNullOrEmpty(NewPowerLE) && !string.IsNullOrEmpty(NewPowerkW) && !string.IsNullOrEmpty(NewEngineSi
        && !string.IsNullOrEmpty(NewDrivingLicence) && !string.IsNullOrEmpty(NewPlaces) && !string.IsNullOrEmpty(NewPr
        && !string.IsNullOrEmpty(NewDeposit) && !string.IsNullOrEmpty(NewTrafficDate) && !string.IsNullOrEmpty(NewLoca
        {
            if (int.TryParse(NewYear, out year) && float.TryParse(NewPowerLE, out Le) && float.TryParse(NewPowerkW, ou
                && float.TryParse(NewEngineSize, out engine) && int.TryParse(NewPlaces, out place) && int.TryParse(New
                && int.TryParse(NewDeposit, out deposit) && DateTime.TryParse(NewTrafficDate, out date))
            {
                if (year <= DateTime.Now.Year && year >= 1800)
                {
                    if (NewLicencePlate.Split('-').First() is string && IsNumber(NewLicencePlate.Split('-').Last()))
                    {
                        string nowdate = date.ToString("yyyy.MM.dd.");
                        var addmotor = new Dictionary<string, object>
                        {
                            {"brand", NewBrand},
                            {"type", NewType},
                            {"licencePlate", NewLicencePlate},
                            {"year", year},
                            {"gearbox", NewGearbox},
                            {"fuel", NewFuel},
                            {"powerLE", Le.ToString(CultureInfo.InvariantCulture)},
                            {"powerkW", kW.ToString(CultureInfo.InvariantCulture)},
                            {"engineSize", engine},
                            {"drivingLicence", NewDrivingLicence},
                            {"places", place},
                            {"price", price},
                            {"deposit", deposit},
                            {"trafficDate", nowdate},
                            {"location", NewLocation},
                            {"image", NewImageName},
                            {"isInService", 0},
                            {"problemComment", ""}
                        };
                        await services.AddMotorAsync(addmotor, NewBrand, NewType, LoggedInUser.Id, LoggedInUser, Image
                        IsAddmotorValid = true;
                    }
                }
            }
        }
    }
}

```

```

    var parameters = new Dictionary<string, object>
    {
        { "LoggedInUser", LoggedInUser }
    };
    await Shell.Current.GoToAsync(nameof(ShowMotorcyclePage), parameters);
}

```

1.4.3.3 Motor kivonása a flottából

Minden motor kártyáján található egy Törlés gomb. Ha a Szerelő erre kattint, akkor a motor kivonásra kerül a flottából. Ez úgy történik, hogy az API (a DELETE metódus segítségével) kitölți a deleted_at mező értékét az aktuális időpontra. Ezt követően a motor többet nem fog megjelenni a motorok listájában, azonban az adatbázisban megmarad. A flottából kivett motor felületen keresztül nem visszaállítható.

```

//Delete Motor to the database
1 reference
public async Task DeleteMotorAsync(int itemId, string motorBrand, string motorType, int CurrentAdminId, AdminModel LoggedAdmin)
{
    try
    {
        var uri = $"http://127.0.0.1:8000/api/DeleteMotor/{itemId}";

        var response = await _httpClient.DeleteAsync(uri);

        if (response.IsSuccessStatusCode)
        {
            string formattedDate = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");
            var log = new LogModel
            {
                command = $"Sikeresen TÖRLTE a {itemId}. Id-jú {motorBrand}-{motorType} a motort",
                date = formattedDate,
                admin_id = CurrentAdminId,
            };
            await adminService.AddLog(log);

            var parameters = new Dictionary<string, object>
            {
                {"LoggedInUser", LoggedAdmin }
            };
            await Shell.Current.GoToAsync(nameof(ShowMotorcyclePage), parameters);
        }
        else
        {
            string responseContent = await response.Content.ReadAsStringAsync();
            await Shell.Current.DisplayAlert("Hiba", $"Nem sikerült a törlés. \n Részletek: {responseContent}", "OK");
        }
    }
    catch (Exception ex)
    {
        string filePath = Path.Combine(FileSystem.AppDataDirectory, "error.txt");
        File.WriteAllText(filePath, ex.ToString());
        throw;
    }
}

```

1.4.3.4 Motor adatainak módosítása

Minden motor kártyáján található egy Módosítás gomb. Ha a Szerelő erre kattint, akkor megnyílik az „EditMotorcyclePage.xaml” oldal. Itt láthatóak az aktuális adatok (pl. márka, rendszám, bérleti ár), amelyeket a Szerelő módosíthat. A Módosítás gombra történő kattintást után az alkalmazás ellenőrzést végez, hasonlóan az új motor felvételéhez. Ha minden rendben van, akkor az API (PUT metódus segítségével) módosítja az adatokat az adatbázisban. Ezt követően az alkalmazás visszatér a „ShowMotorcyclePage.xaml” oldalra.

```

//Update Motor to the database
1 reference
public async Task UpdateMotorAsync(int itemId, Dictionary<string, object> updateData, int CurrentAdminId, int motorId, string type, AdminModel LoggedAdmin)
{
    try
    {
        var uri = $"http://127.0.0.1:8000/api/UpdatesMotor/{itemId}";
        var content = new StringContent(JsonConvert.SerializeObject(updateData), Encoding.UTF8, "application/json");
        var response = await _httpClient.PutAsync(uri, content);
        if (response.IsSuccessStatusCode)
        {
            string formattedDate = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");
            var log = new LogModel
            {
                command = $"Sikeresen FRISSITETTE a {motorId}, {type} a motort",
                date = formattedDate,
                admin_id = CurrentAdminId,
            };
            await adminService.AddLog(log);

            var parameters = new Dictionary<string, object>
            {
                {"LoggedInUser", LoggedAdmin }
            };
            await Shell.Current.GoToAsync(nameof>ShowMotorcyclePage), parameters);
        }
        else
        {
            string responseContent = await response.Content.ReadAsStringAsync();
            await Shell.Current.DisplayAlert("Hiba", $"Nem sikerült a frissítés. \n Részletek: {responseContent}", "OK");
        }
    }
    catch (Exception ex)
    {
        string filePath = Path.Combine(FileSystem.AppDataDirectory, "error.txt");
        File.WriteAllText(filePath, ex.ToString());
        throw;
    }
}

```

1.4.3.5 Motor adatainak megtekintése, szervizbe küldés

Minden motor kártyáján található egy Részletek gomb. Ha a Szerelő erre kattint, akkor megnyílik a „DetailShowMotorcyclePage.xaml” oldal, ahol látható a motor összes adata, a problémái és a képe.

Itt található még a „Szervízbe küldés” gomb, amelyet megnyomva az API (PUT metódussal) módosítja az isInService cella értékét true-ra. Ennek hatására a motor eltűnik a főoldalról, és csak a szervízben levő motorok listájában jelenik meg.

```

//Sending a motorcycle to a service in the database
1 reference
public async Task UpdateServiceMotorAsync(int itemId, Dictionary<string, object> updateData, int CurrentAdminId, string type, AdminModel LoggedAdmin)
{
    try
    {
        var uri = $"http://127.0.0.1:8000/api/UpdatesMotor/{itemId}";

        var content = new StringContent(JsonConvert.SerializeObject(updateData), Encoding.UTF8, "application/json");

        var response = await _httpClient.PutAsync(uri, content);
        if (response.IsSuccessStatusCode)
        {
            string formattedDate = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");
            var log = new LogModel
            {
                command = $"Sikeresen FRISSITETTE a {itemId}, {type} a motort",
                date = formattedDate,
                admin_id = CurrentAdminId,
            };
            await adminService.AddLog(log);

            var parameters = new Dictionary<string, object>
            {
                {"LoggedInUser", LoggedAdmin }
            };
            await Shell.Current.GoToAsync(nameof(ShowMotorcyclePage), parameters);
        }
        else
        {
            string responseContent = await response.Content.ReadAsStringAsync();
            await Shell.Current.DisplayAlert("Hiba", $"Nem sikerült a frissítés. \n Részletek: {responseContent}", "OK");
        }
    }
    catch (Exception ex)
    {
        string filePath = Path.Combine(FileSystem.AppDataDirectory, "error.txt");
        File.WriteAllText(filePath, ex.ToString());
        throw;
    }
}

```

1.4.3.6 Szervizben levő motorok javítása

A szervizben levő motorok az „InServicePage-xaml” oldalon tekinthetők meg, itt az API (GET metódussal) kílistázza azokat a motorokat, amiket a szervizbe küldtek. Láthatóak a motorok főbb adatai (kép, márka, típus, forgalmi lejárási dátuma).

A kiválasztott motor kártyáján a Javítás gombra kattintva megnyílik a „NoteProblamMotorcyclePage.xaml” oldal, ahol szerkeszthető a hibák listája. Itt a szerelő kitörölheti a megjavított hibákat. Ezt követően a „Visszaállítás a flottába” gomb megnyomásának hatására az API (PUT metódussal) módosítja az isInService cella értéke false-re, és a problamComment is frissül. Így a motor visszakerül a kölcsönözhető motorok közé (azaz újra megjelenik a főoldalon, és eltűnik a szervízben levő motorok listájából).

1.4.4 Recepciósok oldalai

Ha belépett admin Recepciós, akkor a következő műveleteket végezheti el (a következő oldalakhoz lesz hozzáférése).

1.4.4.1 Recepciós főoldal

Ha az adott admin pozíciója Recepciós, akkor a rendszer a belépést követően a „SwitchPage.xaml” oldalra viszi, ahol két gomb látható: „Kölcsönzések megtekintése” és „Új felhasználó(k) jogosítvány ellenőrzése”.

1.4.4.2 Kölcsönzések áttekintése

A Kölcsönzések megtekintése a „ShowUsersPage.xaml” oldalon történik. Itt a jövőbeli, illetve a még le nem zárult bérleseket látja a Recepciós (a már lezárult bérlesek nem jelennek meg).

Ehhez az API (GET metódussal) egyszerre lekéri le a users, motors, loans, device_switches, tools táblákat, erre egy külön modell készült.

Minden Kölcsönzés kártyáján megjelennek a kölcsönzés legfontosabb adatai (kölcsönzés azonosító, motor márkája, felhasználó neve, kölcsönzési időtartam).

```

//Get all rental data from the database
1 reference
public async Task<List<UserLoanInfoModel>> GetDataAsync()
{
    try
    {
        var response = await _httpClient.GetStringAsync("http://127.0.0.1:8000/api/usersData");

        var jsonObject = JObject.Parse(response);
        var loansList = new List<UserLoanInfoModel>();

        if (jsonObject["loans"] is JArray loansArray)
        {
            foreach (var item in loansArray)
            {
                if (item["gaveDown"].ToString() == "0")
                {
                    var loan = new UserLoanInfoModel
                    {
                        Id = item["id"].Value<int>(),
                        RentalDate = item["rentalDate"].Value<DateTime>(),
                        ReturnDate = item["returnDate"].Value<DateTime>(),
                        orders_id = item["orders_id"].ToString(),
                        motorId = item["motorcycle"]["id"].Value<int>(),
                        brand = item["motorcycle"]["brand"].ToString(),
                        type = item["motorcycle"]["type"].ToString(),
                        licencePlate = item["motorcycle"]["licencePlate"].ToString(),
                        year = item["motorcycle"]["year"].Value<int>(),
                        gearbox = item["motorcycle"]["gearbox"].ToString(),
                        fuel = item["motorcycle"]["fuel"].ToString(),
                        powerLe = item["motorcycle"]["powerLe"].Value<double>(),
                        powerKw = item["motorcycle"]["powerKw"].Value<double>(),
                        engineSize = item["motorcycle"]["engineSize"].Value<double>(),
                        drivingLicence = item["motorcycle"]["drivingLicence"].ToString(),
                        location = item["motorcycle"]["location"].ToString(),
                        image = item["motorcycle"]["image"].ToString(),
                        problemComment = item["motorcycle"]["problemComment"].ToString(),
                        name = item["user"]["name"].ToString(),
                        email = item["user"]["email"].ToString(),
                        phoneNumber = item["user"]["phoneNumber"].ToString(),
                        drivingLicenceNumber = item["user"]["drivingLicenceNumber"].ToString(),
                        drivingLicenceType = item["user"]["drivingLicenceType"].ToString(),
                        frontDrivingLicenceCard = item["user"]["drivingLicenceImage"].ToString(),
                        backDrivingLicenceCard = item["user"]["drivingLicenceImageBack"].ToString(),

                        ToolsList = new List<Tools>()
                    };

                    if (item["device_switches"] is JArray deviceSwitchesArray)
                    {
                        loan.ToolsList = deviceSwitchesArray
                            .Select(switchItem => switchItem["tool"])
                            .Where(tool => tool != null) // Üres vagy null eszközök kihagyása
                            .Select(tool => new Tools
                            {
                                toolName = tool["toolName"].ToString(),
                                size = tool["size"].ToString()
                            }).ToList();
                    }

                    loansList.Add(loan);
                }
            }
            return loansList;
        }
        else
        {
            return null;
        }
    }
    catch (Exception ex)
    {
        string filePath = Path.Combine(FileSystem.AppDataDirectory, "error.txt");
        File.WriteAllText(filePath, ex.ToString());
        throw;
    }
}

```

1.4.4.3 Kölcsönzés törlése

Azok a kölcsönzések, amelyek keretében még nem vittél el a motort, kitörölhetőek. Az ilyen kölcsönzések kártyáján található egy Törlés gomb, erre kattintva az API (DELETE metódus) segítségével kitörli a loans és device_switches táblából az adott bérletet. Ez végleges törlést jelent, a kitörölt kölcsönzés nem visszaállítható.

```
//Delete rental to the database
1 reference
public async Task DeleteRentalAsync(int itemId, string orederId, int CurrentAdminId, AdminModel LoggedAdmin)
{
    try
    {
        var uri = $"http://127.0.0.1:8000/api/delete-order/{itemId}";
        var response = await _httpClient.DeleteAsync(uri);

        if (response.IsSuccessStatusCode)
        {
            string formattedDate = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");
            var log = new LogModel
            {
                command = $"Sikeresen TÖRLTE a {itemId} Rendelést",
                date = formattedDate,
                admin_id = CurrentAdminId,
            };
            await adminService.AddLog(log);

            var parameters = new Dictionary<string, object>
            {
                {"LoggedInUser", LoggedAdmin }
            };
            await Shell.Current.GoToAsync(nameof>ShowUsersPage), parameters);
        }
        else
        {
            string responseContent = await response.Content.ReadAsStringAsync();
            await Shell.Current.DisplayAlert("Hiba", $"Nem sikerült a törlés. \n Részletek: {responseContent}", "OK");
        }
    }
    catch (Exception ex)
    {
        string filePath = Path.Combine(FileSystem.AppDataDirectory, "error.txt");
        File.WriteAllText(filePath, ex.ToString());
        throw;
    }
}
```

1.4.4.4 Kölcsönzések részleteinek megtekintése

Minden Kölcsönzés kártyáján található egy Részletek gomb, erre kattintva megnyílik a „DetailShowPage.xaml” oldal. Itt minden bérleti adat látható (pl. a bérleti időtartam, a kibérelt motor márkája, a bérző telefonszáma, a bérző jogosítványának a képe).

1.4.4.5 Motor átadása a bérzőnek

A motor részleteit megjelenítő oldalon („DetailShowPage.xaml”) található az Átvette gomb. Erre kattintva az API (PUT metódus segítségével) módosítja a motors táblában a location értékét null-ra. Ez azt jelenti, hogy a motor nincs a telephelyen. Ezt követően az Átvette gomb Disabled lesz, azaz még egyszer nem lehet átadni.

```

//It is called on rental
1 reference
public async Task TakenFromMotorAsync(int itemId, Dictionary<string, object> updateData, int CurrentAdminId, string type, string brand, string username, AdminModel LoggedAdmin)
{
    try
    {
        var uri = $"http://127.0.0.1:8000/api/receiptMotorData/{itemId}";

        var content = new StringContent(JsonConvert.SerializeObject(updateData), Encoding.UTF8, "application/json");

        var response = await _httpClient.PutAsync(uri, content);

        if (response.IsSuccessStatusCode)
        {
            string formattedDate = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");
            var log = new LogModel
            {
                command = $"{username}-nak sikeresen ÁTADTA a {brand} {type} motort",
                date = formattedDate,
                admin_id = CurrentAdminId,
            };
            await adminService.AddLog(log);

            var parameters = new Dictionary<string, object>
            {
                {"LoggedInUser", LoggedAdmin }
            };
            await Shell.Current.GoToAsync(nameof(ShowUsersPage), parameters);
        }
        else
        {
            string responseContent = await response.Content.ReadAsStringAsync();
            await Shell.Current.DisplayAlert("Hiba", $"Nem sikerült az átadás. \n Részletek: {responseContent}", "OK");
        }
    }
    catch (Exception ex)
    {
        string filePath = Path.Combine(FileSystem.AppDataDirectory, "error.txt");
        File.WriteAllText(filePath, ex.ToString());
        throw;
    }
}

```

1.4.4.6 Motor visszavétele

A motor részleteit megjelenítő oldalon („DetailShowPage.xaml”) található a Leadás gomb. Erre kattintva megnyílik a „ProblemReporting.xaml” oldal. Itt egy ellenőrzési lista található, amelynek minden sorát ki kell pipálnia a Recepciósnak, csak akkor lesz visszavezető a motor. Ha bármelyik ellenőrzendő tételel kapcsolatban probléma merült fel, akkor a jobb oldali pipát kipipálva a hiba szövege automatikusan bekerül a szerkeszthető hibalistába (problamComment). Ezt a hibalistát kézzel is ki lehet egészíteni. Ezt követően a Recepciós beállítja a telephelyet, ahol a motor visszavételre került, és megnyomja az „Ellenőrzés befejezése” gombot. Ennek hatására az API (PUT metódus segítségével) módosítja a motor location adatát a megadott helyszínre, a gaveDown adatot true-ra állítja. Ennek hatására a kölcsonzés eltűnik a nyitott kölcsonzések listájából, és visszanavigálunk a „ShowUsersPage.xaml” oldalra.

```

//It is called on return
1 reference
public async Task UpdateRetrieveMotorAsync(int itemId, int motorId, Dictionary<string, object> updateData, int CurrentAdminId, string problems, string brand, AdminModel LoggedAdmin)
{
    try
    {
        var uri = $"http://127.0.0.1:8000/api/retrieveData/{itemId}/{motorId}";

        var content = new StringContent(JsonConvert.SerializeObject(updateData), Encoding.UTF8, "application/json");

        var response = await _httpClient.PutAsync(uri, content);
        if (response.IsSuccessStatusCode)
        {
            string formattedDate = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");

            var log = new LogModel
            {
                command = $"Sikeresen ÁTVETTE és {problems} a motort",
                date = formattedDate,
                admin_id = CurrentAdminId,
            };
            await adminService.AddLog(log);

            var parameters = new Dictionary<string, object>
            {
                {"LoggedInUser", LoggedAdmin }
            };
            await Shell.Current.GoToAsync(nameof(ShowUsersPage), parameters);
        }
        else
        {
            string responseContent = await response.Content.ReadAsStringAsync();
            await Shell.Current.DisplayAlert("Hiba", $"Nem sikerült az átvétel. Részletek: {responseContent}", "OK");
        }
    }
    catch (Exception ex)
    {
        string filePath = Path.Combine(FileSystem.AppDataDirectory, "error.txt");
        File.WriteAllText(filePath, ex.ToString());
        throw;
    }
}

```

1.4.4.7 Új felhasználó jogosítvány ellenőrzése

Ha a Recepciós a főoldalon az „Új felhasználó(k) jogosítvány ellenőrzése” gombot választja, akkor a „ShowAllUserPage.xaml” oldal nyílik meg. Itt látható néhány adat (név, email cím, telefonszám) azokról az ügyfelekről, akik már regisztráltak, de még nem került ellenőrzésre a jogosítványuk. A listát az API (GET metódussal segítségével) kéri le.

Minden egyes új ügyfél kártyáján van egy Részletek gomb, ami átnavigál a „DetailShowUserCheckPage.xaml” oldalra. Ezen az oldalon a Recepciós látja az ügyfél minden adatát és a jogosítvány képét. Ezen az oldalon van egy Jóváhagyás gomb, amelyet megnyomva az API (PUT metódus segítségével) módosítja az aktuális ügyfél drivingLicenceReal cella értékét true-ra. Ettől kezdve az adott felhasználó már tud motort bérálni. Az akció után a recepció automatikusan visszakerül a „ShowAllUserPage.xaml” oldalra.

```
//It is called on validation
1 reference
public async Task UpdateValidationAsync(int itemId, Dictionary<string, object> updateData, string UserName, int CurrentAdminId, AdminModel LoggedAdmin)
{
    try
    {
        var uri = $"http://127.0.0.1:8000/api/user/{itemId}";
        var content = new StringContent(JsonConvert.SerializeObject(updateData), Encoding.UTF8, "application/json");
        var response = await _httpClient.PutAsync(uri, content);

        if (response.IsSuccessStatusCode)
        {
            string formattedDate = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");
            var log = new LogModel
            {
                command = $"Sikeresen JÓVÁHAGYTA a {UserName}-et motort",
                date = formattedDate,
                admin_id = CurrentAdminId,
            };
            await adminService.AddLog(log);

            var parameters = new Dictionary<string, object>
            {
                {"LoggedInUser", LoggedAdmin }
            };
            await Shell.Current.GoToAsync(nameof(ShowAllUserPage), parameters);
        }
        else
        {
            string responseContent = await response.Content.ReadAsStringAsync();
            await Shell.Current.DisplayAlert("Hiba", $"Nem sikerült a művelet. \n Részletek: {responseContent}", "OK");
        }
    }
    catch (Exception ex)
    {
        string filePath = Path.Combine(FileSystem.AppDataDirectory, "error.txt");
        File.WriteAllText(filePath, ex.ToString());
        throw;
    }
}
```

2. Felhasználói dokumentáció

2.1 A felhasználói weboldalak bemutatása

2.1.1 Felhasználói adatok

Weboldalon a Regisztráció gombra kattintva tud az ügyfél regisztrálni, ha eddig nem regisztrált. A regisztráció során elkérjük a bérő teljes nevét, az email címét, az általa megadott jelszavát, a telefonszámát, a jogosítvány azonosító kódját, miket vezethet ezzel a jogosítvánnyal (A1 vagy A2 vagy B) és egy-egy képet a Jogosítvány minden oldaláról.

A regisztráció után kap az ügyfél egy ellenőrző e-mailt és addig nem tud kölcsönözni motort ameddig az ellenőrző e-mailnél rá nem nyomot a megerősítésre. Plusz védelem, hogy a recepciósnak jóvá kell hagynia az ügyfelek által megadott adatokat. Ameddig nem erősíti meg a recepció, hogy minden rendben, addig az ügyfél nem tud bérálni.

Bejelentkezéskor az email-t meg a felhasználó által megadott jelszót kéri el az oldal, ha egyezik az adatbázisban lévő email jelszó párossal a megadott email jelszó páros akkor a bal felső sarokban ott lesz az ügyfél neve, így most már tud motort bérálni.

Bejelentkezés nélkül nem tud bérálni a felhasználó motort mindenkorregisztrálnia kell. Ha a bérlet gombra kattint, az webes látogató és nincs bejelentkezve akkor a weboldal megkéri, hogy jelentkezzen be vagy regisztráljon vagy, hogy várjon egy kis türelemmel amíg meg nem erősítik az adatait.

2.1.2 Felhasználói felület

Ahogy megnyitja az oldalt lát egy nagy képet és hogy milyen weboldal ez, alatta ott van, hogy miért minket a válasszanak, kölcsönzés menete, és hogy miben emelkedünk ki a több rivális céggel szemben.

A weboldal navigációs sávjában van egy **Főoldal gomb**, **Motorok gomb**, **Helyszínek gomb** és egy **Rólunk gomb**. A navigációs sávon helyezkedik el még a **User képe** a jobb sarokban és ha arra rá kattintasz akkor megjelenik a **Bejelentkezés** és a **Regisztrációs gomb**. Ha már bejelentkeztél akkor az előző helyett egy **Profil** valamit egy **Kijelentkezés gombot** látsz, és még a navigációs sávon vannak a mi social mediás eléréseinek is.

Főoldal gomb: Ha rákattint akkor a főoldalra visz át.

Motorok gomb: Ha rákattint akkor át visz egy oldalra, ahol az összes motort megnézheti képpel együtt. A Bérő tud szűrni a fentiekben említett paraméterekre szerint „**Motorok**” alcímben. Az egyes motorokról a márkáját, típusát, teljesítményét (Le, kW), hogy milyen a sebeség váltó, és hogy jelenleg hol tartózkodik a motor, ha nincs ott helyszín akkor egy bérlnél van.

Helyszínek gomb: Ha rákattint akkor át visz egy olyan oldalra, ahol meg lehet nézni a google térkép segítségével, hogy hol vannak a telephelyeink.

Rólunk gomb: Ha rákattint akkor át visz egy olyan oldalra, ahol a mi kamu űstörténetünkéről lehet olvasni.

Bejelentkezés gomb: Ha rákattint akkor át visz egy olyan oldalra, ahol be lehet jelentkezni. A fenti részben „Felhasználói adatok” alcímben megemlíjtük, hogy miket kérünk el a felhasználótól bejelentkezéskor.

Regisztrációs gomb: Ha rákattint akkor át visz egy olyan oldalra, ahol lehet regisztrálni. A fenti részben „Felhasználói adatok” alcímben megemlíjtük, hogy miket kérünk el a felhasználótól regisztráláskor.

Profil gomb: Ha rákattint akkor át visz egy olyan oldalra, ahol látod a rendeléseidet itt is lehet törleni és módosítani a bérleést. Itt látod még az adataidai, amiket módosíthat.

Kijelentkezés gomb: Ha rákattint akkor át visz egy olyan oldalra, ahol ki lehet jelentkezni.

A footerben megtalálhatóak az elérhetőségünk (helyszín, email, telefonszám) valamint itt is vissza tudsz menni a főoldalra a „**Kezdőlap**” linkre kattintva linkre kattintva. Az adatvédelmet és a felhasználói feltételeket itt lehet megnézni, valamint a nyitvatartásunkat.

2.1.3 Foglalás menete

A motor bérleséhez első lépésként a felhasználónak regisztrálnia kell vagy be kell jelentkeznie a weboldalon. Ezután a navigációs sávban található **Motorok** fülre kattintva tudja megkezdeni a keresést. Itt a szűrők segítségével beállíthatja a kívánt bérlesi időszakot, a motormárkát, az évjáratot, meghajtástípust és egyéb szempontokat – ahogyan ezt a „**Motorok**” szakaszban korábban részleteztük.

Miután kiválasztotta a neki tetsző motort, a motor adatlapján belül a **Részletek** fülön tud rákattintani a **Bérles** gombra. Ekkor választható ki a bérles kezdő- és záró dátuma egy dátumválasztó segítségével. A rendszer automatikusan kiszámolja a bérles időtartamát napokban.

Ezután a felhasználó kérhet védfelszerelést is, ami nem kötelező, de javasolt. A választható felszerelések:

- **Bukósisak** (méret és mennyiség kiválasztható, maximum 2 db)
- **Protektoros ruha vagy kabát** (maximum 2 db)
- **Motoros csizma** (maximum 2 pár)

A felszerelésekhez napi bérleti díj és kaució tartozik:

- Bukósisak: 5 000 Ft/nap, 20 000 Ft kaució/sisak
- Protektoros ruha: 10 000 Ft/nap, 30 000 Ft kaució/db
- Csizma: 8 000 Ft/nap, 25 000 Ft kaució/pár

A felszerelések kiválasztása és a dátum megadása után megjelenik egy összegző oldal, ahol a felhasználó részletesen láthatja a foglalás összes adatát: a kiválasztott motort, a bérlet időtartamát, a felszerelések díját, az esetleges kedvezményeket (3–6 nap között 20%, 7 nap felett 30%), a kaució összegét, valamint a végleges fizetendő díjat.

Ha minden rendben talál, rákattinthat a **Bérlet megerősítése** gombra. Ezzel a foglalás véglegesítésre kerül, és a felhasználó kap egy egyedi rendelési azonosítót a következő formátumban: ORD-XXXXXXX.

2.2 A mobilapplikáció bemutatása

2.2.1 Mobil application

A mobilos felületen az ügyfél láthatja az összes rendelését, és a még meg nem történt bérleseit is, amit törölhet, viszont, ha egy napon a rendelés kezdetéig akkor már nem tudja visszamondani a bérleést. A felhasználónak elsőnek be kell jelentkeznie email-jelszó párossal, majd ezután láthatja a saját rendeléseit.

Az oldal elején láthatóak az ügyfél adatai, mint például a telefonszáma vagy az email címe, ha lejjebb görget akkor meg láthatja a bérleseit, ami tartalmazza például a rendelési azonosítóját.

Motorkölcsönzésre nincsen MÉG lehetőség, de megtekinthetőek az elérhető motorok “Foglalj most!” gombra kattintva, mint egy katalógus.

Későbbiekben ezekhez hozzá lehet tenni bérleti funkciókat, de ezek még jelenleg munkálatok alatt folynak.

Felhasználói Profil

Személyes Adatok

Név: Szabó Bálint

Email: baliko.sef@gmail.com

Telefonszám: 06202151263

Vezetői engedély száma: at234567

Vezetői engedély típusa: AM

Jogosítvány Előlap Jogosítvány Hátlap



Aktuális Kölcsönzések

Kölcsönzési Adatok

Rendelési azonosító: ORD-4IRFCL8K

Kölcsönzés kezdete: 2025-05-20

Visszahozás dátuma: 2025-05-23

Státusz: Megvalósult bérlet

Motor Kép



Törles

Motorkölcsönzésre nincsen MÉG lehetőség, de megtekinthetők az elérhető motorok "Foglalj most!" gombra kattintva, mint egy katalógus.

Későbbiekben ezekhez hozzá lehet tenni bérleti funkciókat, de ezek még jelenleg munkálatok alatt folynak.

2.3 Adminisztrációs alkalmazás bemutatása

2.3.1 Adminisztrátori alkalmazás

Az adminisztrátori alkalmazás szerveroldali komponense egy központi számítógépen fut, amelyhez az adminok a saját gépeikről csatlakoznak egy kliens alkalmazás segítségével. Hárrom munkakört különböztetünk meg az adminisztrátori alkalmazásban: Szuperadmin,

Szerelő és Recepciós. minden egyes munkakör más oldalakat lát, vagyis például egy Recepciós nem tud felvenni új alkalmazottat.

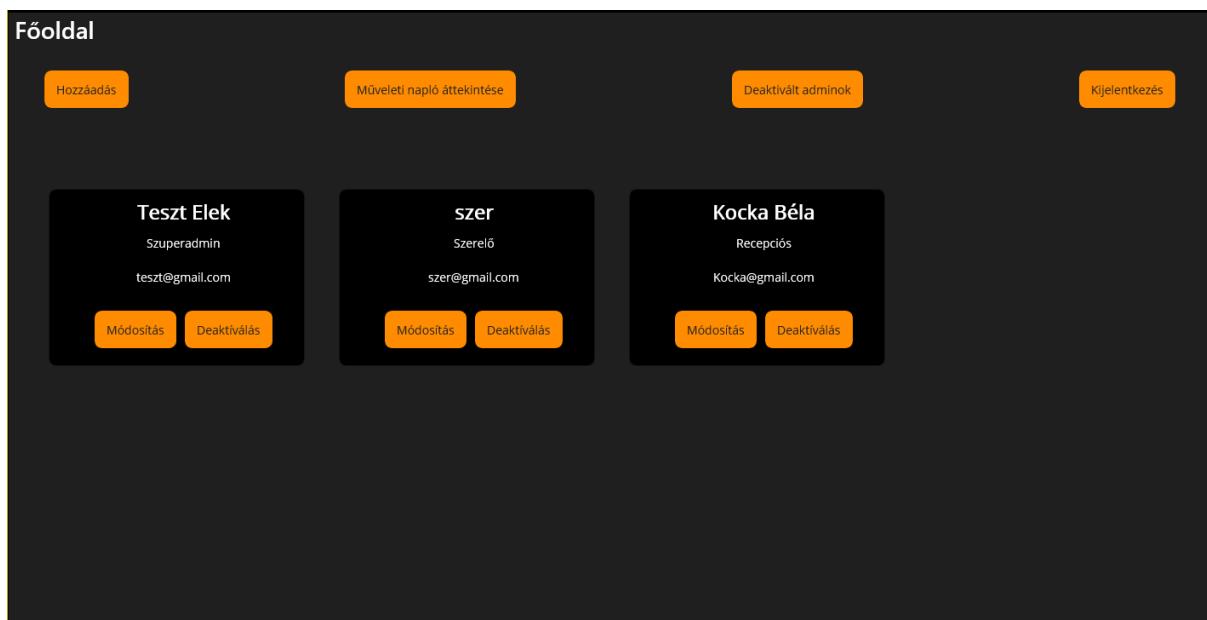
A **Szuperadminok** foglalkoznak az Adminisztrátori alkalmazás felhasználóival. Ők tudnak felvenni új embereket az alkalmazásba, tudják módosítani a szerepköreiket, a nevüket, az email címüket és a jelszavakat, valamint deaktiválni tudják az alkalmazottakat, és vissza is tudják állítani az felhasználójukat. Ezen kívül meg tudják nézni a logokat, és ezáltal láthatják, hogy az egyes alkalmazottakat miket csináltak a felületen.

A **Szerelők** a motorokkal foglalkoznak. Hozzá tudnak adni a flottához új motorokat úgy, hogy beviszik az adatokat a motorról, mint például a rendszámát, márkáját. Ki tudják venni a flottából a motorokat, és módosíthatják is az adataikat, mint például a teljesítmény vagy akár a bérleti ár. A Szerelők, a részletek oldalon meg tudják nézni a motor hibáit és adatait, és itt tudják berakni a szervizbe. Ha szervizben van a motor, akkor az ügyfelek nem tudják kibérelni.

A **Recepciósok** az ügyfelek motor bérlésével és adataegyeztetésével foglalkoznak.

2.3.2 Szuperadmin felület

Amikor a Szuperadmin belép a felületére, az első dolog, amit látni fog, az összes adminisztrátor listája, akik már regisztráltak. Ezen kívül a következő menüpontok jelennek meg: Hozzáadás, Műveleti napló megtekintése, Deaktivált adminok és Kijelentkezés.



Minden motor kártyáján két gomb szerepel: Módosítás és Deaktiválás.

- Adminok módosítása:** minden egyes admin kártyán van egy Módosítás gomb, ami átvíz egy másik oldalra, ahol látható lesz az összes adata (kivéve a jelszót), amit lehet módosítani, ha a módosítás oldalon rányomsz a Módosítás gombra akkor módosítja az adott admin adatait.

Felhasználó kezelés

The screenshot shows a user management interface. At the top, there are three input fields: 'szer' (placeholder), 'szer@gmail.com' (placeholder), and an empty field. Below these is a dropdown menu labeled 'Poszt' (Post) with 'Szerelő' selected. At the bottom are two buttons: 'Módosítás' (Edit) and 'Vissza' (Back).

- Deaktiválás:** Erre a gombra kattintva lehet az adott admint deaktiválni. Ezt követően eltűnik a főoldalról, és csak a Deaktivált adminok listájában jelenik meg.

A felső menüpontok választása esetén a következők történnek:

- Új admin hozzáadása:** Itt új adminisztrátorokat adhat hozzá. Ehhez meg kell adnia az admin teljes nevét, egy egyedi email címet és egy megfelelő jelszót (kis- és nagybetűk, számok, speciális karakterek). Emellett ki kell választania a pozíóját (Szuperadmin, Szerelő vagy Recepciós). Ha minden adatot megadott, akkor a "Hozzáadás" gombra kattintva a rendszer ellenőrzi, hogy minden rendben van-e. Ha igen, hozzáadja az új adminisztrátort.

Admin hozzáadása

The screenshot shows an 'Add Admin' form. It has three input fields: 'Admin neve' (placeholder), 'Admin email címe' (placeholder), and 'Admin jelszava' (placeholder). Below these is a dropdown menu labeled 'Poszt' (Post) with 'Szuperadmin' selected. At the bottom are two buttons: 'Hozzáadás' (Add) and 'Mégsem' (Not now).

- Műveleti napló áttekintése:** Itt megtekintheti, hogy az egyes adminisztrátorok milyen műveleteket végeztek el a rendszerben.

Műveleti napló

Vissza

teszt@gmail.com

Műveleti esemény(ek):

Sikeresen MÓDOSÍTOTTA az szer admint.
2025-05-04 08:35:48

Sikeresen deaktiválta JUCIKA-ot(-ét) az 1 számú Id-val
2025-05-04 08:34:13

Sikeresen deaktiválta Gipsz Jakab-ot(-ét) az 1 számú Id-val
2025-05-04 08:33:55

Sikeresen MÓDOSÍTOTTAA az JUCIKA admint.
2025-05-04 08:33:43

Sikeresen MÓDOSÍTOTTAA az Kocka Béla admint.
2025-05-04 08:32:48

Sikeresen MÓDOSÍTOTTAA az Gipsz Jakab admint.
2025-05-04 08:32:22

Sikeresen MÓDOSÍTOTTAA az szer admint.
2025-05-03 20:42:28

Sikeresen MÓDOSÍTOTTAA az szer admint.
2025-05-03 20:24:17

Sikeresen HOZZÁADTA szer Recepciós admint.
2025-05-03 20:20:45

Sikeresen HOZZÁADTA JUCIKA Recepciós admint.
2025-05-03 20:05:54

- Deaktivált adminok:** Itt megtekintheti a deaktivált adminokat, és ha szükséges, visszaállíthatja őket.

Deaktivált adminok

Vissza

Gipsz Jakab

Szerelő

Gipsz@gmail.com

Visszaállítás

JUCIKA

Recepciós

Juci@gmail.com

Visszaállítás

- Kijelentkezés:** Ha befejezte a munkát, itt tud kijelentkezni a Szuperadmin.

2.3.3 Szerelő felület

A Szerelő felülete egy kicsit másképp működik. A főoldal alapértelmezetten kilistázza az elérhető (kölcsönözhető) motorokat, ezek fölött pedig a következő menüpontok lesznek láthatóak: Hozzáadás, Szervizben, Kijelentkezés.

Motorok

Hozzáadás Szervizben Kijelentkezés

5  Aprilia SR 160 2028.12.18 Részletek Módosítás Törlés	6  Aprilia Tuono 125 2020.06.17 Részletek Módosítás Törlés	7  Aprilia Tuono 660 2028.05.22 Részletek Módosítás Törlés	8  BMW C 400 GT 2025.10.10 Részletek Módosítás Törlés
---	--	---	---

A motorok háttere két színű lehet: zöld vagy piros. A zöld azt jelenti, hogy még érvényes a forgalmija, a piros háttér jelentése, hogy lejárt. minden motornál megjelenhet jobb felül egy „🔧” ikon, ami azt jelenti, hogy valami baja van a motornak és érdemes szervízbe vinni.

Minden motor kártyáján három gomb szerepel: Részletek, Módosítás, Törlés

- **A Részletek oldalon** láthatóak a motor problémái, adatai és a motor képe. lehet a motort a szervizbe küldeni, így eltűnik a főoldalról a motor és a szerviz oldalon jelenik meg.

Motor részletes adatai

Vissza

Motoron történő javítások: [vak] (A motor részletes adatainak előtérben álló része)

Motor Neve: BMW - Concept 101
Rendszáma: DJB-831
Évjárat: 2022
Sebességváltó típusa: Manualis (6 fokozatú)
Üzemanyag típusa: B
Teljesítménye: 158,2 LE - 115,5 kW
Motor méret (cm³): 1649
Milyen jogosítvánnyal lehet használni: A2
Ülésök száma: 2
Bérleti ára: 32767
Bérleti kauciós ára: 145200
Forgalom lejáratú időpontja: 2027. 08. 14. 0:00:00

Szervizbe küldés

- Motorok adatainak módosítása:** A Módosítás gombra kattintva egy másik oldalra jutunk, ahol látható lesz a motor összes adata, ezeket lehet módosítani. Ha végeztünk, akkor az oldal alján található Módosítás gomb segítségével menthetjük el az adatokat.

Motor kezelése

Márka:	Aprilia
Típus:	RSV4 Factory
Rendszám:	PNB-672
Gyártási év:	2010
Sebességváltó típus	Manualis (6 fokozatú)
B:Benzin / E:Elektromos	B
Teljesítmény (LE):	177,6
Teljesítmény (kW):	129,6
Motor méret (cm ³):	999,6
Jogosítvány típusa	A2
Ülések száma:	2
Ár (Ft/nap):	32767
Kaució (Ft):	133200
Forgalmi lejárat időpontja:	2027. 04. 01. <input type="button" value="Módosítás"/>
Helyszín	1126 Budapest, Böszormányi út 38. <input type="button" value="Módosítás"/>
<input type="button" value="Mentés"/> <input type="button" value="Vissza"/>	

- Motor törlése:** A Törlés gombra kattintva a motor kikerül a flottából, azaz többet nem látszik a felületen, de az adatbázisban bent marad.

A felső menüpontok választása esetén a következők történnek:

- **Hozzáadás:** Erre a gombra kattintva egy új oldalra jutunk, ahol lehetőség van új motor felvételeire. A megjelenő űrlapon meg kell adni a motor adatait, például a márkáját, bérleti árat és lóerejét. A rendszer ellenőri, hogy minden adatot helyesen adott-e meg, és ha igen, hozzáadja a motort a flottához.

Új motor felvétele

Márka:
Pl.: BMW, Harley-Davidson

Típus:
Pl.: Gxt-15

Rendszám:
Pl.: asd-123 vagy aa.aa-12

Gyártási év:
Pl.: 2015

Sebességváltó

E: Elektromos v. B: Benzines

Teljesítmény (LE):
Pl.: 120,5

Teljesítmény (kW):
Pl.: 88,3

Motor méret (cm³):
Pl.: 1250

Jogosítvány

Férőhelyek száma:
Pl.: 2

Ár (Ft/nap):
Pl.: 8000

Kaució (Ft):
Pl.: 30000

Forgalmi lejárat időpontja: 2025-05-04

Location

Kép hozzáadása
Kép kiválasztása

Hozzáadás Mégsem

- **Szervizben:** Itt láthatóak azok a motorokat, amelyek szervizbe kerültek.

Szerviz

Vissza

3



Aprilia
RS 125
2026.11.08

Javítás

Módosítás

Törlés

4



Aprilia
RSV4 Factory
2027.04.01

Javítás

Módosítás

Törlés

A szervizben levő motorok kártyáján három gomb szerepel: Javítás, Módosítás, Törlés. A Módosítás és a Törlés ugyanúgy működik, mint ahogyan fentebb már ismertetésre került.

- **Javítás:** Erre a gombra kattintva egy újabb oldalra jutunk, ahol megtekinthetők a motor problémái, amelyek szerkeszthetők, azaz itt dokumentálhatja a szerelő, ha kijavított egy hibát. A módosítások elmenthetők a Mentés gomb segítségével, ekkor a motor még a szervizben marad. Ha a motor javítása befejeződött, akkor a szerelő a Visszaállítás a flottába gombot nyomja meg, ennek hatására a motor visszakerül a flottába, és már nem fog látszani a szervizben levő motorok között.

Problémabejegyzés

Hibalista szerkesztése

hibás a kipufogó

Visszaállítás a flottába

Mentés

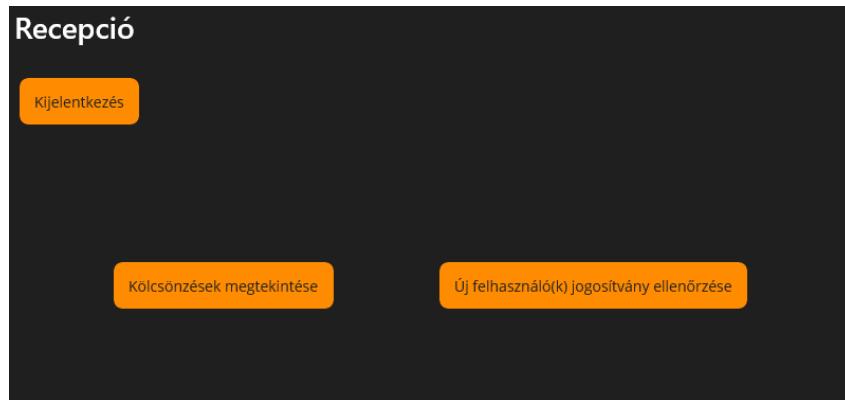
Vissza

- **Kijelentkezés:** Ha végezett, itt tud kijelentkezni a felületről.

2.3.4 Recepciós felület

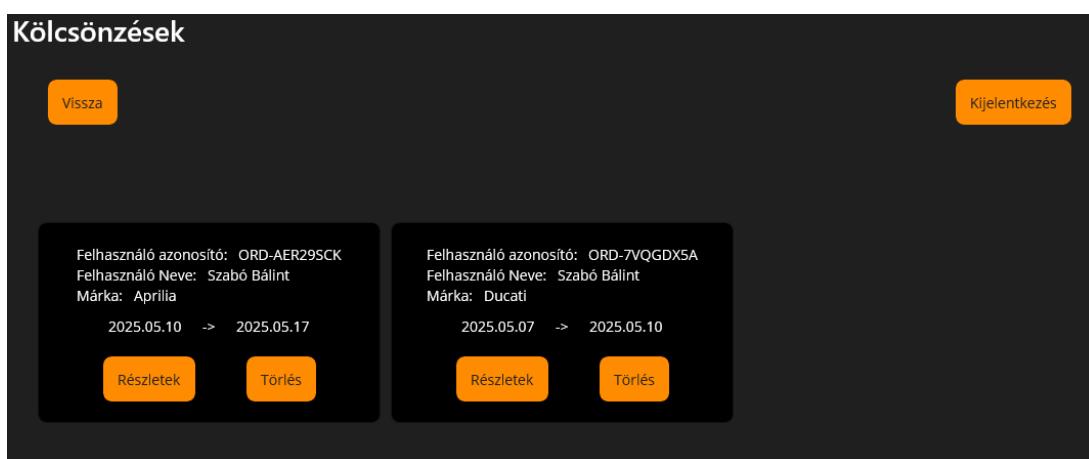
A Recepciós felületén két fő gomb található:

- Új felhasználó jogosítványának ellenőrzése:** Itt tudja ellenőrizni és jóváhagyni az ügyfél adatait.
- Kölcsönzések megtekintése:** itt tudja kezelní a kölcsönzés adatait, például rögzíteni a motor átadását vagy visszavételét.



Kölcsönzések megtekintése:

A Kölcsönzések megtekintése gombra kattintva egy újabb oldal nyílik meg, ahol az aktuális kölcsönzések listája szerepel. Csak a jövőbeli, illetve a még le nem zárult bérletek láthatóak itt (a már lezárt bérletek nem jelennek meg). Keresni lehet a bérlet azonosítója vagy az ügyfél neve alapján. A Recepciós ezeket a bérleteket kezelheti, és akár törölni is tud bérleteket, például, ha az ügyfél nem jött el a lefoglalt motorért.



Minden kölcsönzés kártyáján két gomb található: Részletek és Törlés.

- Törlés:** Erre a gombra kattintva a kölcsönzés törlésre kerül. A törlés nem visszaállítható.
- Részletek:** Erre a gombra kattintva megjelennek az ügyfél és a motor adatai. Itt minden bérlesi adat látható (pl. a bérlesi időtartam, a kibérelt motor márkája, a bérő telefonszáma). A bérő jogosítványának a képe is megtekinthető, ha nem annyira látható a kép akkor, ha rákattint a képre megnyílik egy külön oldalon a kép nagyban és plusz funkció, hogy a képet lehet forgatni.

A Recepciós itt ellenőrizheti például, hogy az ügyfél rendelkezik-e a motor vezetéséhez szükséges jogosítvánnyal. Az ügyfél a helyszínen fizeti ki a bérlet díját a Recepciósnak, a motor csak ezt követően kerül átadásra.

Bérlet részletei

Motor hibák:



Felhasználó név: Szabó Bélint

Email: baliko.sef@gmail.com

Telefonszáma: 06202151263

Jogosítvány száma: CZ123456

Jogosítvány típusa: A1



Kibérelt Motor: Aprilia - RS 125

Motor Rendszáma: GZR-419

Milyen jogosítvánnyal vezethető: A2

2025.05.10 -> 2025.05.17



- Sisak: XL
- Protektoros Ruhá: S
- Protektoros Ruhá: L
- Cipő: 42

Átvette

Leadás

Vissza

Ezen az oldalon három gomb található: Átvette, Leadás és Vissza.

- **Átvette:** Erre kattintva egy újabb oldalra jutunk, ezt a Recepciós akkor használja, amikor a motort ténylegesen átadja az ügyfélnek. Ekkor a motor helyszín értéke nullára változik, jelezve, hogy az ügyélnél van. Ezt a gombot a Recepciós csak egyszer tudja megnyomni, így nem adhatja át többször a motort.
- **Leadás:** Erre kattintva egy újabb oldalra jutunk, ezt a Recepciós akkor használja, amikor a motort az ügyfél visszahozza a kölcsönzőbe. Ezen a külön oldalon megjelenő checklist alapján ellenőrzi a motor állapotát. Ha bármilyen hibát talál, azt feljegyzi, és a kár levonja a kaucióból. A hibák a motor visszavételével kapcsolatos adatok között kerülnek tárolásra.

Motor ellenőrzés

Motor Visszavételei Ellenőrzőlista

1. Általános állapot

- | | |
|--|--|
| <input type="checkbox"/> A motor tiszta és sérülésmentes | <input type="checkbox"/> Hiba Felirása |
| <input type="checkbox"/> Nincsenek karcolások, törések, horpadások | <input type="checkbox"/> Hiba Felirása |
| <input type="checkbox"/> minden alkatrész hiánytalan (tükrök, indexek, lámpák) | <input type="checkbox"/> Hiba Felirása |

2. Kerekek és gumiik

- | | |
|---|--|
| <input type="checkbox"/> A gumik állapota megfelelő | <input type="checkbox"/> Hiba Felirása |
| <input type="checkbox"/> A gumik légnyomása rendben van | <input type="checkbox"/> Hiba Felirása |
| <input type="checkbox"/> A felnik sértetlenek | <input type="checkbox"/> Hiba Felirása |

3. Fékek és felfügggesztés

- | | |
|--|--|
| <input type="checkbox"/> A fékek megfelelően működnek | <input type="checkbox"/> Hiba Felirása |
| <input type="checkbox"/> A fékbetétek és tárcsák állapota jó | <input type="checkbox"/> Hiba Felirása |
| <input type="checkbox"/> A teleszkópek nem szivárognak | <input type="checkbox"/> Hiba Felirása |

4. Világítás és elektronika

- | | |
|---|--|
| <input type="checkbox"/> Az összes lámpa és index működik | <input type="checkbox"/> Hiba Felirása |
| <input type="checkbox"/> A kürt működik | <input type="checkbox"/> Hiba Felirása |
| <input type="checkbox"/> A műszerfalon nincs hibaüzenet | <input type="checkbox"/> Hiba Felirása |

5. Motor és mechanika

- | | |
|--|--|
| <input type="checkbox"/> Az olajszint megfelelő | <input type="checkbox"/> Hiba Felirása |
| <input type="checkbox"/> Nincs olaj-, üzemanyag- vagy hűtőfolyadék-szivárgás | <input type="checkbox"/> Hiba Felirása |
| <input type="checkbox"/> A motor könnyen indul és normálisan működik | <input type="checkbox"/> Hiba Felirása |

6. Tartozékok és dokumentáció

- | | |
|---|--|
| <input type="checkbox"/> A kulcsokat hiánytalanul visszahozták? | <input type="checkbox"/> Hiba Felirása |
|---|--|

Problémák:

Helyszín

Ellenőrzés befejezése

Új felhasználó jogosítvány ellenőrzése:

Ha a Recepciós a főoldalon erre a gombra kattint, akkor egy újabb oldalra jut, ahol azokat az ügyfeleket, akiket még nem hagytak jóvá.

Jogosítvány ellenőrzés

Vissza Kijelentkezés

Felhasználó Neve: Test User
Email: test@gmail.com
Telefonszáma: 0620222222

Részletek

Felhasználó Neve: Szabó Bálint
Email: baliko.sef@gmail.com
Telefonszáma: 06202151263

Részletek

Itt a kiválasztott ügyfél kártyáján a Részletek gombra kattintva megtekinthetőek az ügyfél adatai, köztük a jogosítvány fényképe is. A jóváhagyáshoz a Recepciós egy külső közhites adatbázis segítségével ellenőrzi a megadott jogosítványt és az azon szereplő adatokat. Ha a jogosítvány képe esetleg nem látszana elég jól, akkor a képre kattintva az megnyílik egy külön ablakban nagyobb méretben, illetve plusz funkció, hogy a képet lehet forgatni. Ha minden jónak lát, akkor jóváhagyja az ügyfelet. Az ügyfél innentől kezdve bérelhet motort. (Azt, hogy a jogosítvány valóban a kölcsönhő ügyfél-e, a motor átvételekor kerül ellenőrzésre.)

Felhasználói adatok ellenőrzése

Vissza

Felhasználó név: Szabó Bálint
Email: baliko.sef@gmail.com
Telefonszáma: 06202151263

Jogosítvány száma: CZ123456
Jogosítvány típusa: A1



jóváhagyás



3. Tesztelési dokumentáció

Az elkészített program megfelelőségét Unit tesztekkel és black box tesztekkel ellenőriztük.

3.1 Szerveroldal tesztelése

3.1.1 API Teszt

S. No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1.	GET /userData	N/A	Status 200, JSON list	Status 200, JSON list	Postman	Pass	A kölcsönzés összes adata megjelenik
2.	GET /logs	N/A	Status 200, JSON list	Status 200, JSON list	Postman	Pass	Az összes admin és az ahoz tartozó logok is megjelentek
3.	GET /allUsers	N/A	Status 200, JSON list	Status 200, JSON list	Postman	Pass	A összes User megjelenik
4.	GET /allMotors	N/A	Status 200, JSON list	Status 200, JSON list	Postman	Pass	A összes motor megjelenik
5.	POST /admins	{"name": "John Doe", "email": "johndoe@gmail.com", "password": "yourPassword", "jobStatus": 1}	Status 200, JSON list	Status 200, JSON list	Postman	Pass	Hozzáadta az adatbázishoz
6.	POST /AddMotor	{"brand": "Toyota", "type": "T-23", "licencePlate": "ATI-123", "year": 2020, "gearbox": "Automata", "fuel": "B", "powerLe": 120, "powerkW": 88, "engineSize": 2000, "drivingLicence": "B", "places": 1, "price": 20000, "deposit": 5000, "trafficDate": "2025-05-01", "location": "1161 Budapest, Rákosi út 88.", "isInService": 0, "problemComment": null}	Status 200, JSON list	Status 200, JSON list	Postman	Pass	Hozzáadta az adatbázishoz
7.	POST /AddLog	{"command": "Sikeresen HOZZÁADTA Bélát adminokhoz.", "date": "2020-02-02", "admin_id": 1}	Status 200, JSON list	Status 200, JSON list	Postman	Pass	Hozzáadta az adatbázishoz
8.	DELETE /DeleteMotor/1	Auth token	Status 200, JSON list	Status 200, JSON list	Postman	Pass	Kitörölte a 1 ID motort
9.	DELETE /DeactiveAdmin/1	Auth token	Status 200, JSON list	Status 200, JSON list	Postman	Pass	Kitörölte a 1 ID admint

10.	PUT /retrieveData/1/3	{„location”:1012 Budapest, Logodi utca 34. „problamComment”:A kürt nem működik „gaveDown”:1}	Status 200, JSON list	Status 200, JSON list	Postman	Pass	Sikeresen módosult az adat
11	PUT /user/1	{„drivingLicenceReal”:0}	Status 200, JSON list	Status 200, JSON list	Postman	Pass	Sikeresen módosult az adat
12	PUT /admins/1	{„name”:John Doe „email”:johndoe@gmail.com „password”:yourPassword „jobStatus”:1}	Status 200, JSON list	Status 200, JSON list	Postman	Pass	Sikeresen módosult az adat
13	PUT /UpdatesMotor/11	{„brand”:Toyota „type”:T-23 „licencePlate”:ATI-123 „year”:2020 „gearbox”: Automata „fuel”:B „powerLe”:120 „powerkW”:88 „engineSize”:2000 „drivingLicence”:B „places”:1 „price”:20000 „deposit”:5000 „trafficDate”:2025-05-01 „location”:1161 Budapest, Rákosi út 88. „isInService”:0 „problamComment”:null}	Status 200, JSON list	Status 200, JSON list	Postman	Pass	Sikeresen módosult az adat

3.2 A felhasználói weboldalak tesztelése

3.2.1 Regisztráció

S. No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	Fill out registration form	Name: John Doe, Email: john.doe@example.com, Password: password123, Phone: 123456789	Form fields are correctly filled.	Form fields are correctly filled.	Chrome	Pass	Form data accepted.

2	Submit the registration form	Click on the "Register" button	User registered successfully in the system.	User registered successfully in the system.	Chrome	Pass	Registration completed successfully.
3	Verify database entry	Check database for user entry	User data exists in the database.	User data exists in the database.	N/A	Pass	User record saved correctly in the database.
4	Verify file uploads	Check the storage folder for uploaded driving license images.	Files are saved in the storage directory.	Files are saved in the storage directory.	N/A	Pass	Driving license images uploaded successfully.

PASS | RegistrationTest
✓ create user with valid data saves to database

3.2.2 Bérlés hozzáadása

S. No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	Create a loan record in the database	users_id: 1, motorcycles_id : 2, rentalDate: '2025-01-01', returnDate: '2025-01-05', other metadata	Loan record is created successfully in the database.	Loan record is created successfully in the database.	N/A	Pass	Loan record created with the provided details.
2	Create a device switch record	loans_id: Generated loan ID, tools_id: 1	Device switch record is created successfully.	Device switch record is created	N/A	Pass	Device switch linked to the created loan record.

				successfull y.			
3	Verify loan record	Assert database has a record with id: Loan ID and users_id: 1	Loan record exists in the database.	Loan record exists in the database.	N/A	Pass	Loan record verified successfully in the database.
4	Verify device switch record	Assert database has a record with loans_id: Loan ID and tools_id: 1	Device switch record exists in the database.	Device switch record exists in the database.	N/A	Pass	Device switch record verified successfully.

PASS Tests\Feature\MotorcycleRentalTest
✓ loan and device switch creation

3.2.3 Felhasználó profil módosítás

S. No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	Fetch user with ID 2	User ID: 2	User is successfully fetched from the database.	User is successfully fetched from the database.	N/A	Pass	User with ID 2 exists in the database.
2	Simulate authentication	Acting as the user with ID 2	User is authenticated successfully.	User is authenticated successfully.	N/A	Pass	Authentication simulated successfully.
3	Mock storage for file uploads	Mock storage for public disk	Storage is mocked successfully.	Storage is mocked successfully.	N/A	Pass	Storage for file uploads prepared.
4	Prepare request data	name: "Új Név", email: "uj@example.com", phoneNumber: "22222222", file inputs for license images.	Request data is prepared successfully.	Request data is prepared successfully.	N/A	Pass	Request created with updated user data.
5	Execute update method	Call UserController::update with prepared request.	Controller processes the update request successfully.	Controller processes the update request	N/A	Pass	User update logic executed successfully.

				successfully.			
6	Verify updated user data	Assert database for updated values: name, email, phoneNumber.	User data in the database matches updated values.	User data in the database matches updated values.	N/A	Pass	Database reflects updated user details.

PASS UserManageTest
✓ update method updates existing user with id 2

3.2.4 Eszköz hozzáadása a foglaláshoz

S. No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	Fetch loan order with ID 1	Loan ID: 1	Loan order is successfully fetched from the database.	Loan order is successfully fetched from the database.	N/A	Pass	Loan order is available for tool addition.
2	Validate loan order exists	Assert non-null loan order	Loan order exists in the database.	Loan order exists in the database.	N/A	Pass	Loan order validation successful.
3	Fetch tools with IDs 1, 6, 11	Tool IDs: 1 (helmet), 6 (protector suit), 11 (shoes)	Tools are successfully fetched from the database.	Tools are successfully fetched from the database.	N/A	Pass	All required tools exist.
4	Validate tool existence	Assert non-null tools for IDs 1, 6, 11	Tools exist in the database.	Tools exist in the database.	N/A	Pass	All tools validated for addition.
5	Add each tool to the loan order	Post request to addToolToOrder route with tool IDs 1, 6, and 11.	Each tool is linked to the loan order in the database.	Each tool is linked to the loan order in the database.	N/A	Pass	Tools added successfully.

6	Verify database updates	Assert device_switches contains records with loans_id=1 and respective tools_id.	Records exist in device_switches table.	Records exist in device_switches table.	N/A	Pass	Database reflects tool additions.
7	Verify redirection and success message	Response redirect with session message: "Új eszköz hozzáadva a rendeléshez ."	Redirection and success message confirmed.	Redirection and success message confirmed.	N/A	Pass	Proper feedback provided to the user.

PASS Tests\Feature\AddToolToOrderTest
 ✓ it adds multiple tools to an existing order

3.2.5 Eszköz törlése a foglalásból

S. No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	Fetch loan order with ID 2	Loan ID: 2	Loan order is successfully fetched from the database.	Loan order is successfully fetched from the database.	N/A	Pass	Loan order is available for tool deletion.
2	Fetch tool with ID 1	Tool ID: 1	Tool is successfully fetched from the database.	Tool is successfully fetched from the database.	N/A	Pass	Tool is available for deletion.
3	Fetch device_switch linking tool to loan order	Tool ID: 1, Loan ID: 2	A device_switch record is found in the database.	A device_switch record is found in the database.	N/A	Pass	Relationship between tool and loan verified.
4	Delete the device_switch record	Perform delete() on the fetched device_switch record.	The device_switch record is removed from the database.	The device_switch record is removed from the database.	N/A	Pass	Tool successfully removed from the loan.
5	Verify device_switch record is deleted	Assert absence of record in device_switches table.	The device_switch record is no longer present.	The device_switch record is no longer present.	N/A	Pass	Database reflects the deletion.

PASS Tests\Feature\DeleteToolFromOrderTest
 ✓ deletes a tool

3.2.6 Foglalás törlése

S . No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	Fetch loan order with ID 1	Loan ID: 1	Loan order is successfully fetched from the database.	Loan order is successfully fetched from the database.	N/A	Pass	Loan order exists for deletion.
2	Verify associated device_switch records exist	Loan ID: 1	Each device_switch record exists in the database.	Each device_switch record exists in the database.	N/A	Pass	Relationships verified.
3	Delete the loan order	Loan ID: 1	Loan order is removed from the loans table.	Loan order is removed from the loans table.	N/A	Pass	Loan order deletion successful.
4	Verify associated device_switch records deleted	Loan ID: 1	All related device_switch records are removed.	All related device_switch records are removed.	N/A	Pass	Database reflects cascading deletions.
5	Verify redirection and success message	Route: /userProfile	User is redirected with a success message in the session.	User is redirected with a success message in the session.	N/A	Pass	User feedback provided.

PASS Tests\Feature\DeleteOrderTest
 ✓ it deletes an existing order