

ASZTALI- ÉS WEBES SZOFTVERFEJLESZTÉS, ADATBÁZIS-KEZELÉS FELADATSOR

ÁLTALÁNOS TUDNIVALÓK

- Készítse el az alábbi alkalmazásokat, amelyekkel elvégezhetők az alábbi feladatok!
- Az alkalmazások elkészítéséhez tetszőleges fejlesztői környezetet, illetve programozási nyelvet használhat!
- Segítségül használhatók:
 - o a gépre telepített offline help rendszerek
 - o internetkapcsolat, amely használható:
 - általános keresésre;
 - online dokumentáció elérésére;
 - projekt keretek létrehozására, csomagok telepítésére.
 - Mindenfajta kommunikáció, vagy meg nem engedett segédanyag letöltése szigorúan tilos!
- **Kiadott források:**
 - o cars.sql – a „cars” adatbázistábla szerkezete és kezdeti tartalma
 - o kepek.zip – a feladat megoldásához szükséges képek
- **Beadandó:** az összes projekt, a megoldások során létrejött kimeneti állományok, illetve a megoldás során létrehozott adatbázis exportja.
 - o Mindezeket egyetlen tömörített fájlban töltsse fel Teams feladat beadásaként!
 - o Az állomány neve szoftvervizsga2022proba_sajatnev_osztaly legyen!
 - o A beadott alkalmazások futtatható állapotúak legyenek!
 - o A hibás vagy hiányos részeket kommentben hagyja a kódban, de jelezze, hogy az is a megoldás része!
 - o A gyorsabb értékeléshez az elkészített programot futtatható állapotban hagyja megnyitva a számítógépén!

FELADATLEÍRÁS

Egy autókölcsönző autóit nyilvántartó rendszert kell elkészítenie, amely autók, ill. a kölcsönzések adatait tartja nyilván.

A rendszer három fő komponensből kell álljon:

- egy webes backend alkalmazásból, amely REST API-n keresztül biztosít hozzáférést az adatokhoz;
- egy böngészőben futó kliens alkalmazásból, amely a backend alkalmazást használja;
- egy desktop/konzolos hibrid alkalmazásból.

Az autók listáját az alábbi „cars” adattábla definiálja:

- id: egész szám, az autó azonosítója, elsődleges kulcs, automatikusan kap értéket
- license_plate_number: szöveg, az autó rendszáma
- brand: szöveg, az autó márkája
- model: szöveg, az autó típusa
- daily_cost: egész szám, az autó napidíja forintban
- created_at: timestamp, az autó nyilvántartásba vételének ideje
- updated_at: timestamp, a rekord legutóbbi módosítása az adatbázisban

A backend és a desktop/konzolos alkalmazások ugyanazt a közös adatbázist használják.

Az alkalmazás csak belső, zárt hálózaton lesz elérhető, ezért autentikációt nem kell megvalósítani.

1. feladat Backend alkalmazás

A feladatot egy webes programozási nyelvben kell megvalósítani. Ajánlott egy MVC keretrendszer, ill. egy hozzá tartozó ORM keretrendszer használata (pl. PHP/Laravel/Eloquent, C#/ASP.NET/Entity Framework).

- Készítsen egy üres projektet a választott backend keretrendszer segítségével. A projekt neve legyen **carrentalbackend**! Töltse be a **cars.sql** fájl tartalmát az adatbázis-kezelőbe!
- Készítse el a kölcsönzések időtartamát nyilvántartó **rentals** adattáblát, az alábbi oszlopokkal:
 - **id**: egész szám, a kölcsönzés azonosítója, elsődleges kulcs
 - **car_id**: egész szám, a hivatkozott autó azonosítója, idegen kulcs (a cars.id mezőre hivatkozik)
 - **start_date**: dátum, a kölcsönzés kezdete
 - **end_date**: dátum, a kölcsönzés vége

Amennyiben a választott keretrendszer megköveteli, a tábla tartalmazhat még szükséges oszlopokat (pl. created_at stb.)

Az adattáblát adatbázismigráció segítségével hozza létre!

- Hozza létre az adatbázis táblákhoz tartozó modell osztályokat! (Car és Rental)
- A rentals táblát töltse fel véletlenül generált teszt-adatokkal, legalább 15 rekorddal! Ezt a backend keretrendszer seed-elés (vagy ekvivalens) funkciójával tegye meg!
- Készítse ez az alábbi API végpontokat!

Minden végpont JSON adatformában adja vissza a kimenetet.

Hiba esetén a hiba okát jelezze:

- A HTTP státusz kóddal, valamint
- Egy JSON objektum segítségével szövegesen is
 - o A keretrendszer által generált hiba-válaszok is megfelelők, amennyiben a feltételeknek megfelelnek.

- **GET /api/cars**

Adja vissza az összes autó alábbi 5 adatát: id, license_plate_number, brand, model, daily_cost

Az eredmény egy objektumokból álló lista legyen. Egy lehetséges megoldás:

```
{
  "data": [
    {
      "id": 1,
      "license_plate_number": "ABC-123",
      "brand": "Ford",
      "model": "Escort",
      "daily_cost": "23000"
    },
    {
      "id": 2,
      "license_plate_number": "PLT-123",
      "brand": "Audi",
      "model": "A8",
      "daily_cost": "28000"
    },
    {
      "id": 3,
      "license_plate_number": "SZF-123",
      "brand": "Honda",
      "model": "Civic",
      "daily_cost": "17000"
    }
  ]
}
```

- **POST /api/cars**

Hozzon létre egy új autót.

A kérés törzse egy JSON objektum, amely tartalmazza az alábbi mezőket: license_plate_number, brand, model, daily_cost. Pl.:

```
{
  "license_plate_number": "HEL-666",
  "brand": "Peugeot",
  "model": "Rifter",
  "daily_cost": "20000",
}
```

A végpont ellenőrizze, hogy a bemeneti adatok megfelelők-e:

- Minden mező megadása kötelező
- A daily_cost csak pozitív egész szám lehet

Validációs hiba esetén adjon vissza egy JSON objektumot, amely leírja a hiba okát, valamint egy megfelelő 4xx-es státusz kódot.

Siker esetén adja vissza az új autót leíró JSON objektumot (l. GET /api/cars végpontnál leírtakat), amiben szerepeljen az adatbázis-kezelő által generált id is! A státusz kód a **201 Created** legyen.

- **POST /api/cars/{car}/rent**

Autó kölcsönzése: foglaljon le egy autót egy hétre, azaz hozzon létre egy új Rental rekordot, ahol:

- A kezdődátum az aktuális dátum
- A kölcsönzés vége a jelenlegi dátumhoz képest egy hét
- Az autó azonosítója az URL-ben szereplő ID.

A kérésnek nincs törzse, a {car} paraméter pedig egy egész szám, amely egy autó azonosítóját jelenti.

Ha nincs ilyen azonosítójú autó, a végpont ezt **404 Not Found** státusz kóddal jelezze.

Ha a autó már foglalt (azaz már létezik foglalás az adott autóra, amelynek az intervallumába beleesik az aktuális dátum), akkor a végpont ezt jelezze **409 Conflict** HTTP státusz kóddal, valamint a JSON kimeneten jelezze szövegesen is a hiba okát.

Siker esetén JSON formátumban jelezze a kölcsönzés kezdetét és végét:

```
{
  "id": 43,
  "car_id": 3,
  "start_date": "2022-04-10",
  "end_date": "2022-04-17"
}
```

2. feladat Frontend alkalmazás

A feladatot JavaScript programozási nyelvben (vagy JavaScript-re forduló nyelvben) kell megvalósítani, egy frontend keretrendszer segítségével (pl. Vue.js, React, Angular). CSS keretrendszer (pl. Bootstrap, Tailwind) használata megengedett.

A feladat teljeskörű elkészítéséhez szükség van az 1. feladatban elkészített backend API végpontokra. Amennyiben valamelyiket nem tudta elkészíteni, a frontend alkalmazásban ettől függetlenül hívja meg a végpont URL-jét, az alkalmazás azonban dolgozhat tovább teszt adatokkal.

Az alkalmazás Egyoldalas Alkalmazás (Single Page Application) legyen, vagyis egyetlen funkció se járjon teljes oldal-újrátöltéssel vagy böngésző navigációval.

A feladat elkészítése során, ahol lehetséges, használjon szemantikus HTML tag-eket CSS class-ok és id-k helyett!

- a) Készítsen egy üres projektet a választott frontend keretrendszer segítségével. A projekt neve legyen **carrentalfrentend!**
- b) Az alkalmazás betöltésekor kérdezze le az /api/cars végpont segítségével az autók adatait. Az így lekért autók alábbi adatait jelenítse meg:
 - Rendszám
 - Márka
 - Modell
 - Napidíj
 - Az autó képe

A képeket a mellékelt **kepek.zip** tömörített fájlban találja, ezeket másolja egy olyan mappába, amely a frontend alkalmazás számára elérhető. A kép megnevezése minden esetben **márka_modell.png** formájú pl.: egy Ford Focus esetén ford_focus.png

Az adatok megjelenítésekor az alábbi szempontokat vegye figyelembe:

 - Az autók nevei HTML címsorként szerepeljenek!
 - Reszponzivitás: desktop nézetben három, tablet nézetben kettő, mobil nézetben egy autó adata jelenjen meg soronként!
 - Az autók adatai vizuálisan különüljenek el egymástól (pl. szegély vagy térköz segítségével)!
 - A megjelenítésre egy mintát talál a feladatsor végén.
- c) A táblázat alatt jelenítsen meg egy autó felvételi űrlapot, amely segítségével a szükséges 4 adat megadható (új kép feltöltését nem kell megvalósítani), valamint egy „Új autó” feliratú gombot. A gombra kattintáskor:
 - Próbáljon meg létrehozni egy új autót a megadott adatokkal, az /api/cars végpont használatával!
 - Amennyiben a backend alkalmazás validációs hibát jelez, ezeket jelenítse meg a felhasználónak!

A hibaüzenetet formázza meg úgy, hogy a hibaüzenet jellege egyértelmű legyen!

 - Sikeres létrehozás esetén töltse újra a táblázatot (hogy a legfrissebb adatokkal dolgozhasson), és az űrlapot állítsa alaphelyzetbe.
 - A bemeneti mezők típusai legyenek az adott adattípusnak megfelelőek!
- d) Az autók listáját egészítse ki „Kölcsönzés” feliratú gombokkal, amely minden autó kártyájában/cellájában jelenjen meg! A gombra kattintva hívja meg az /api/cars/{car}/rent végpontot. Siker esetén a gomb alatt jelenjen meg egy „Sikeres kölcsönzés!” felirat, ellenkező esetben pedig írja ki a backend által visszaadott hibaüzenet.
- e) Egészítse ki az alkalmazást:
 - Egy fejléccel, amely tartalmazza:
 - o Az alkalmazás megnevezését: Petrik Autókölcsönző (ez legyen HTML címsor)
 - Ez legyen a teljes oldal címe is!
 - o Egy vízszintes navigációs sávot, amely két linket tartalmazzon:
 - Új autó felvétele – görgessen le a „Új autó” űrlaphoz
 - Petrik honlap – a <https://petrik.hu/> weboldalra mutasson
 - Egy lábléccel, amelyben szerepeljen az alkalmazás készítőjének (azaz az Ön) neve.

[Új autó felvétele](#) [Petrik honlap](#)

Petrik Autókolcsönző

IOA-699

Márka: Ford

Modell: Chile

Napidíj: 24000 Ft



Készítette: Ifj. Minta Vizsga

ABY-945

Márka: Ford

Modell: CX727

Napidíj: 24000 Ft



NTJ-242

Márka: Ford

Modell: Evos

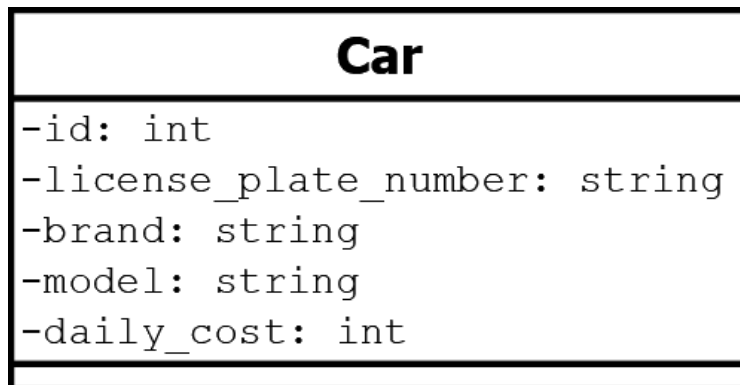
Napidíj: 20000 Ft



3. feladat Konzolos alkalmazásrész, az adatszerkezet kialakítása

A feladatot C# vagy Java programozási nyelvben kell megvalósítani. A megoldáshoz szükséges az 1. feladatban létrehozott és kiegészített adatbázis.

- Hozzon létre egy új projektet **CarRental-desktop** néven!
- Hozzon létre egy **Car** osztályt az autók adatainak kezeléséhez az alábbi osztálydiagramm alapján:



- Az adattagokhoz készítsen get property-ket, vagy getter metódusokat!
 - Hozzon létre a **Car** osztályban paraméteres konstruktort, amely a paraméterek értékével inicializálja az adattagokat!
 - Amennyiben későbbi feladatok megoldásához szükséges, úgy bővítse az osztályt megfelelő adattagokkal és metódusokkal.
- Hozzon létre egy **Statisztika** osztályt a konzolos feladatok elvégzéséhez!
 - A program indításakor amennyiben `--stat` parancssori argumentum lett megadva úgy a konzolos alkalmazásrész induljon el.
 - o Amennyiben nem tudja kezelni a parancssori argumentumokat úgy a konzolos alkalmazásrész számára külön projektet hozzon létre **CarRental-desktop-cli** néven
 - Az osztály rendelkezzen egy `cars` nevű adattaggal, amely egy **Car** típusú objektumokat tartalmazó lista, amely lehetővé teszi a kölcsönző összes autójának a kezelését.
 - Írjon a **Statisztika** osztályban függvényt, amely beolvassa az adatbázisban lévő autókat, és a beolvasott adatok alapján feltölti a „cars” listát a autók.
 - Ha nem sikerül kapcsolódni az adatbázishoz, a program adjon hibaüzenetet, és a program futása szakadjon meg!
 - Hozzon létre a **Statisztika** osztályban konstruktort vagy statikus függvényt, amely végrehajtja a beolvasást, majd elvégzi a további részfeladatokat.
 - Hozzon létre eljárásokat és függvényeket a Statisztika osztályban az alábbi részfeladatok elvégzéséhez. Az eredményt írja ki a konzolra a mintának megfelelően.
 - Határozza meg a 20.000 Ft-nál olcsóbb napidíjú autók számát.
 - Döntse el, hogy szerepel-e az adatok között 26.000 Ft-nál drágább. napidíjú autó
 - Határozza meg és írja ki a legdrágább napidíjú autó adatait.
 - Határozza meg és írja ki márká szerinti csoportosítva az autók számát.
 - Kérjen be a konzolról egy rendszámot. Határozza meg, hogy az adott autó napidíja nagyobb-e mint 25.000 Ft. Ha a megadott rendszámmal nem szerepel autó, akkor „Nincs ilyen autó” üzenet jelenjen meg.
 - Minta:

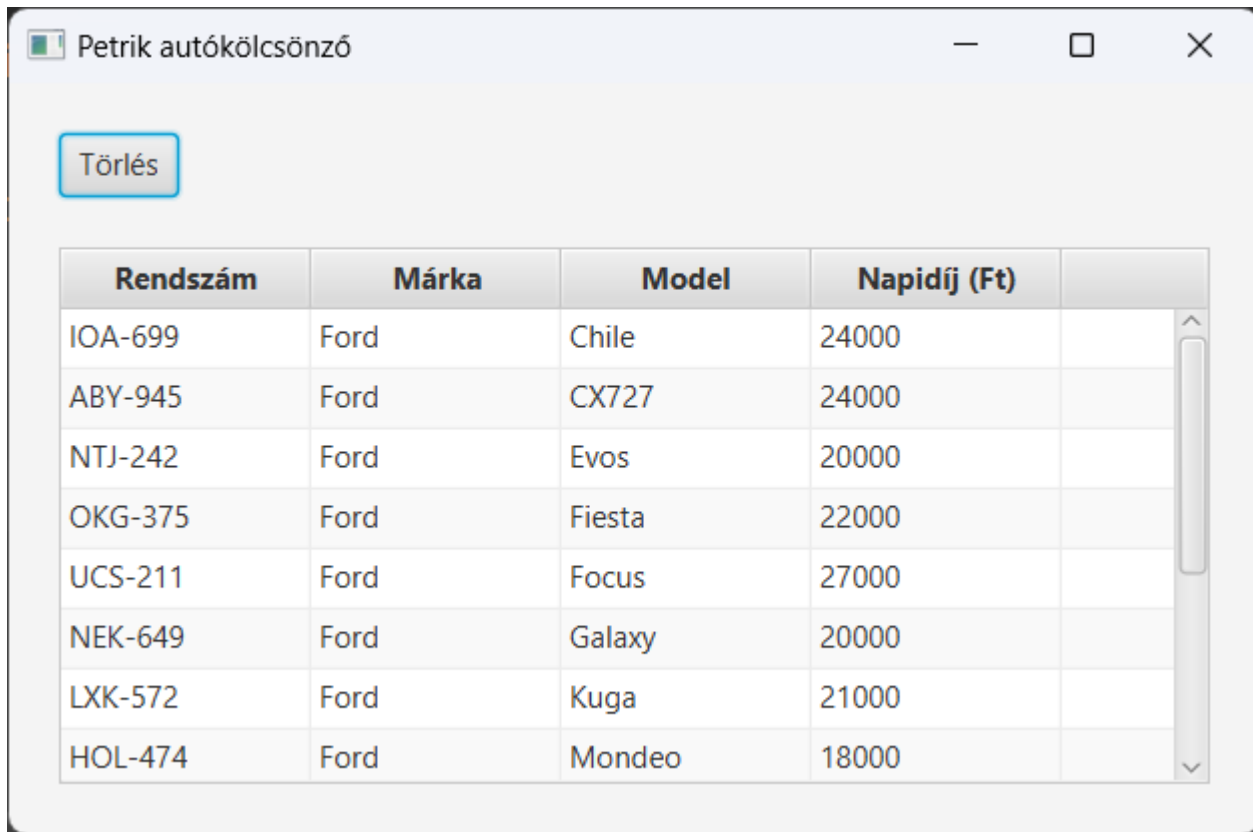
```

20.000 Ft-nál olcsóbb napidíjú autók száma: 2
Van az adatok között 26.000 Ft-nál drágább napidíjú autó
Legdrágább napidíjú autó: UCS-211 - Ford - Focus - 27000 Ft
Autók száma:
    Renault: 1
    Ford: 9
    Peugeot: 1
Adjon meg egy rendszámot: AUO-561
A megadott autó napidíja nem nagyobb mint 25.000 Ft
  
```

4. feladat Grafikus alkalmazásrész – autók listázása és törlése

A megoldást a 3. feladat folytatásaként kell elvégezni, vagyis a két feladat végeredménye egyetlen projekt legyen!

- a) A projekt grafikus megjelenítést végző osztályában hozza létre a felület elemeit úgy, hogy az alábbi mintához hasonló megjelenítést tegyen lehetővé!



- Az ablak bal felső sarkában helyezzen el egy gombot „Törlés” felirattal
- Helyezzen el a gomb alá egy tároló komponens, amelybe a autók adatait táblázatos formában tudja listázni.
 - o A megjelenítésre szolgáló komponensnek nem kötelező rendelkeznie fejléccel.
- b) Az alkalmazásrész indulásakor töltse fel a listát az adatbázisban lévő autók adataival.
 - A listázáshoz használja fel az előző feladatban létrehozott **Car** osztályt.
 - Ha nem sikerül kapcsolódni az adatbázishoz, a program felugró ablakban adjon hibaüzenetet. A felugró ablak bezárásakor a teljes program álljon le.
- c) Tegye lehetővé az autók törlését!
 - A „Törlés” gombra kattintva a listából kiválasztott autó kerüljön eltávolításra az adatbázisból.
 - Amennyiben nincs autó kiválasztva akkor felugró ablakban jelenjen meg „Törléshez előbb válasszon ki autót” üzenet.
 - Ha ki lett választva autó, akkor a törlés előtt jelenjen meg egy megerősítő ablak „Biztos szeretné törölni a kiválasztott autót?” felirattal.
 - A törlést csak akkor hajtsa végre, ha a felhasználó a felugró ablakon megfelelő gombra kattintott.
 - A törlés sikerességéről vagy sikertelenségéről adjon visszajelzést. Sikertelen törlés esetén megfelelő hibaüzenetet jelenítsen meg.
 - A sikeresen eltávolított autó a listából is kerüljön törlésre.