# Autóbérlési Weboldal Projektdokumentáció

## 1. Bevezetés

### 1.1 Projekt célja

Egy vidéki autóbérléssel foglalkozó vállalat forgalma alacsony, és szeretné bővíteni marketing tevékenységét. Az oldal célja egy modern, felhasználóbarát online platform létrehozása, amely segíti az ügyfeleket a megfelelő bérautó kiválasztásában és bérlésében.

### 1.2 Alkalmazás komponensei

Az alkalmazás 3 fő komponensből áll, amelyek együttműködnek:

* Backend: Egy ASP.NET-el megvalósított RESTful API, melyen keresztül egyszerűen lehet hozzáférni az adatbázishoz
* Felhasználói Komponens: Egy statikus, HTML, CSS és Javascript használatával megvalósított weboldal, melyen keresztül a kliensek egyszerűen választhatnak maguknak autót.
* Adminisztrációs komponens: Egy PHP-val megvalósított weboldal, amelyen keresztül a weboldal üzemeltetői egyszerűen módosíthatják a weboldalon szereplő adatokat.

### 1.3 Technológiai stack

#### **HTML (HyperText Markup Language)**

A HTML az alapvető nyelv, amelyet a weboldalak struktúrájának létrehozására használnak. Meghatározza az oldalon található elemeket, például szövegeket, képeket, űrlapokat és linkeket. A HTML segítségével az oldalak különböző böngészőkben jeleníthetők meg egységes módon.

***Bootstrap***

A Bootstrap egy nyílt forráskódú CSS keretrendszer, amelyet a Twitter fejlesztői hoztak létre, és mára az egyik legnépszerűbb front-end fejlesztési eszközzé vált. Elsődleges célja, hogy leegyszerűsítse a reszponzív és modern weboldalak fejlesztését, miközben egységes megjelenést biztosít a különböző böngészők és eszközök között.

#### **CSS (Cascading Style Sheets)**

A CSS felelős a weboldalak megjelenéséért és stílusáért. Lehetővé teszi az elemek színének, méretének, elrendezésének és animációinak meghatározását. Segít a reszponzív design kialakításában, amely biztosítja, hogy a weboldal különböző eszközökön megfelelően jelenjen meg.

#### **JavaScript**

A JavaScript egy dinamikus programozási nyelv, amely interaktív funkciókat biztosít a weboldalak számára. Segítségével valósíthatók meg például animációk, felhasználói események kezelése, vagy adatok aszinkron betöltése az API-ból. Az ES6 verzió és az újabb JavaScript fejlesztések lehetővé teszik a modern és hatékony kódírást.

#### PHP

A PHP egy általános szerveroldali szkriptnyelv dinamikus weblapok készítésére. Az első szkriptnyelvek egyike, amely külső fájl használata helyett HTML oldalba ágyazható. A kódot a webszerver PHP feldolgozómodulja értelmezi, ezzel dinamikus weboldalakat hozva létre.

#### **ASP.NET Core**

Az ASP.NET Core egy nyílt forráskódú, keresztplatformos webalkalmazás-fejlesztési keretrendszer, amely lehetővé teszi a dinamikus weboldalak és API-k fejlesztését. Fő előnyei közé tartozik a gyors teljesítmény, a könnyű skálázhatóság és a különböző platformokon való futtatási lehetőség (Windows, Linux, macOS). Az MVC (Model-View-Controller) architektúra támogatásával könnyen fejleszthető és karbantartható alkalmazások készíthetők.

#### **MariaDB**

A MariaDB egy nyílt forráskódú relációs adatbázis-kezelő rendszer, amely a MySQL továbbfejlesztett verziója. Gyors, megbízható és skálázható adatbázis-megoldás, amely támogatja az SQL lekérdezéseket és a tranzakciókezelést. Az autóbérlő rendszerben az ügyfelek és az autók adatainak tárolására és kezelésére használatos.

#### **Entity Framework Core**

Az Entity Framework Core (EF Core) egy objektum-relációs leképező (ORM) az ASP.NET Core számára, amely lehetővé teszi a fejlesztők számára, hogy az adatbázist objektumokkal kezeljék anélkül, hogy közvetlen SQL-lekérdezéseket kellene írniuk. Az EF Core segítségével az adatok gyorsan és hatékonyan lekérdezhetők és módosíthatók, miközben a rendszer támogatja a különböző adatbázis-kezelő rendszereket.

#### **RESTful API**

A REST (Representational State Transfer) egy architekturális stílus, amelyet API-k tervezésére és fejlesztésére használnak. A RESTful API-k lehetővé teszik az adatok könnyű elérését és kezelését HTTP protokollon keresztül. A Rent A Car projektben az API felelős a frontend és a backend közötti adatátvitelért, biztosítva a CRUD (Create, Read, Update, Delete) műveleteket az autók és ügyfelek adataihoz.

## 

## 2. Backend API

### 2.1 Bevezetés

A backend rész egy RESTful API-ként van megvalósítva. Ez egyszerű és hordozható adatforgalmat tesz lehetővé, amely a jövőbeli fejlesztés és bővítés érdekében jelentős. Ezek mellett külső, nem általunk fejlesztett szolgáltatásokon keresztül is elérhető, amely a jövőben nagyobb forgalmat biztosíthat.

### 2.2 Használt technológiák

#### ASP.NET

Az ASP.NET egy elterjett és kifejlett backend keretrendszer. Kiforrott, sok funkciót tartalmaz és gyors iterációt tesz lehetővé, amik miatt ezt véltük a megfelelő választásnak. A “minimal api” komponse egyszerűvé tette a végpontok kialakítását.

#### Entity Framework Core

Az Entity Framework Core (EF Core) lehetővé tette az adatbázis gyors és biztonságos létrehozását. Mivel a C# osztályokból tökéletes adatbázis sémát tudott létrehozni, nem kellett a hibás összekötés miatt aggódni.

#### MariaDB

A MariaDB megengedte a megszokott MySQL interfész használatát, azonban nyílt forráskódú keretek között.

#### Pomelo

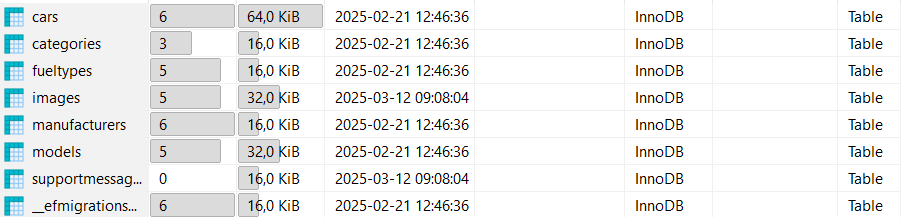
A Pomelo egy EF Core bővítmény könyvtár amely lehetővé tette hogy MariaDB, illetve MySQL adatbázishoz csatlakozzunk MSSql helyett.

#### MinimalApis.Extensions

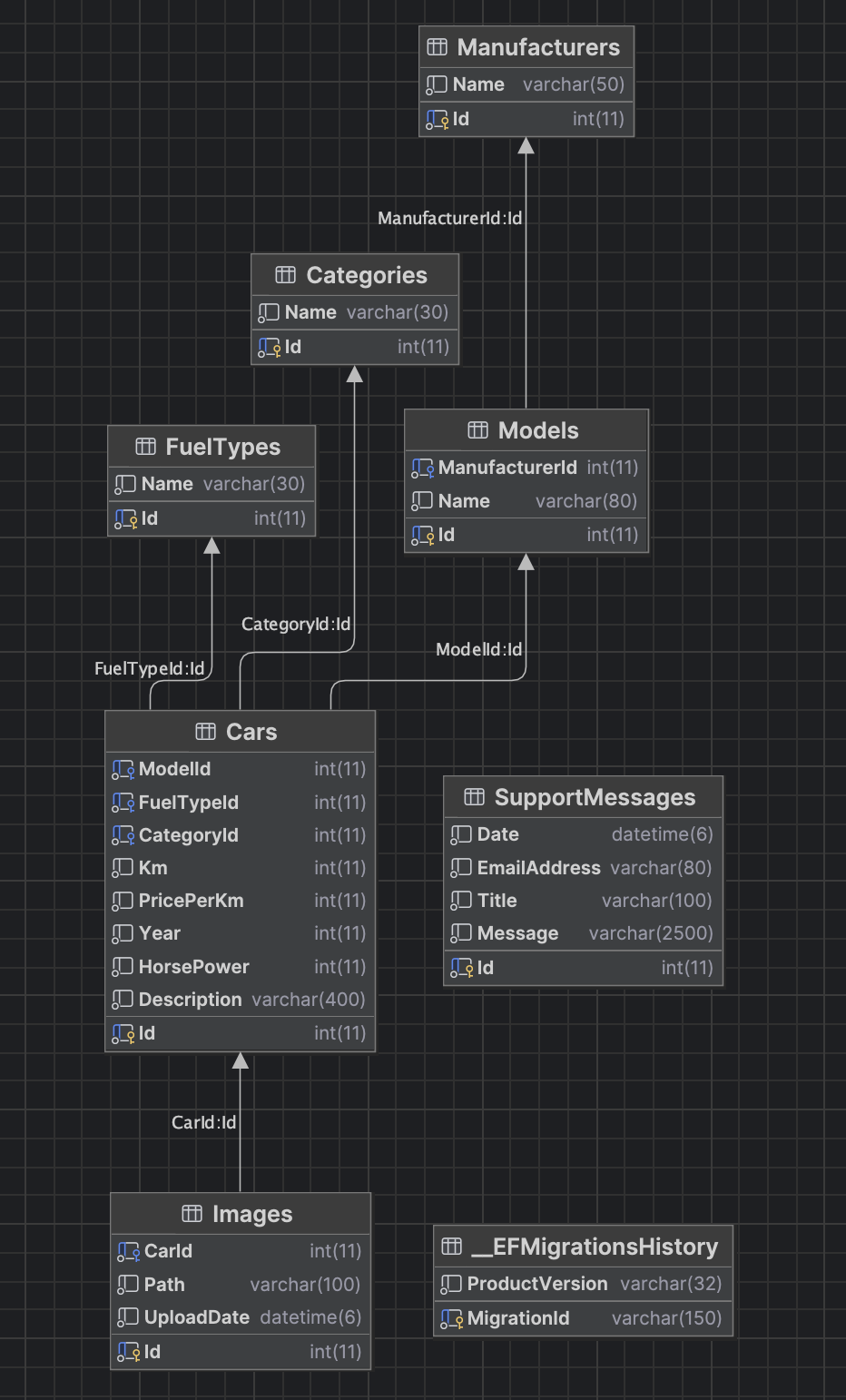
Ez a könyvtár néhány kényelmi funkcióval bővítette a “minimal apis” osztályokat, ezek közül legfontosabb a bemenő adatok vizsgálata volt.

### 2.3 Adatbázis struktúra

Az adatbázis az alábbi főbb táblákat tartalmazza:



* A ‘Cars’ tábla az egyes autókat és a hozzájuk tartozó egyéni adatokat tartalmazza.
* A ‘Categories’ tábla az autók kategóriáját, mint például sport vagy luxus, tartalmazza.
* A ‘FuelTypes’ tábla az elérhető üzemanyag fajtákat, mint benzin vagy dízel, tartalmazza
* Az ‘Images’ tábla a feltöltött képekhez tartozó adatokat tartalmazza. Maguk a képek máshol, fájlként vannak feltöltve, ez a tábla arra helyre mutat.
* A ‘Manufacturers’ tábla a különböző márkákat tartalmazza.
* A ‘Models’ tábla az egyes autómodelleket tartalmazza. Ezeket a tárhely optimalizálása és az egyszerűbb szűrés miatt tároljuk ilyen módon.
* A ‘SupportMessages’ tábla üzeneteket tartalmaz. Ez az applikáció jelenlegi formájában még nincs implementálva.
* Az ‘\_efmigrations’ tábla az Entity Framework Core használatához szükséges

Az alábbi diagram az adatbázis modelljét részletesebben leírja:

### 2.4 Végpontok

Az API az alábbi végpontokat valósítja meg. Alap beállítások szerint ezek a <http://localhost:8080>/ URL-en keresztül érhetőek el. Ezt magán a projekten belül, illetve valós futtatás esetén webszerverrel lehet változtatni. Minden adattípushoz 5 végpont van megvalósítva. 2 GET, 1 POST, 1 PUT és 1 DELETE.

Az alábbi táblázat leírja a elvégezhető műveleteket. A [típus] helyettesíthető a következő adattípusokkal: cars, models, fuel\_types, categories, manufacturers, images.

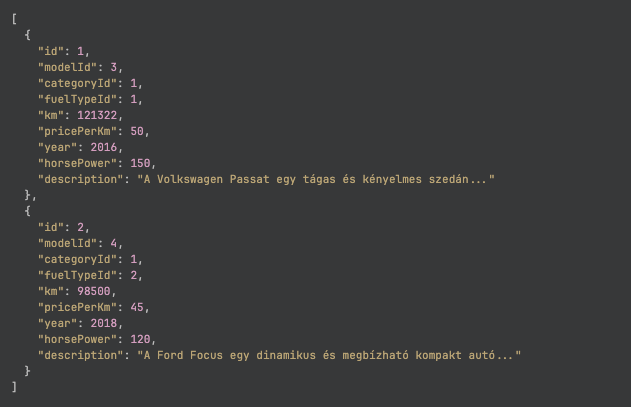
| Művelet | HTTP | Végpont | Leírás |
| --- | --- | --- | --- |
| Lekérdezés (lista) | GET | /[típus] | Az adott típus összes elemét visszaadja |
| Lekérdezés (id) | GET | /[típus]/{id} | Az {id} kulcsú elemet adja vissza |
| Hozzáadás | POST | /[típus] | Hozzáad egy új elemet |
| Frissítés | PUT | /[típus]/{id} | Az {id} elemet lecseréli egy újra |
| Törlés | DELETE | /[típus]/{is} | Kitörli a megadott elemet |

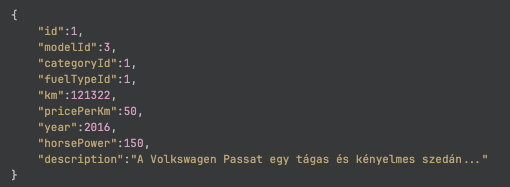
### 

### 2.5 API Használata

#### Lekérdezés

A lekérdezések minden esetben ugyanúgy működnek. Példa:  
A “GET /cars” a követkető választ adja:

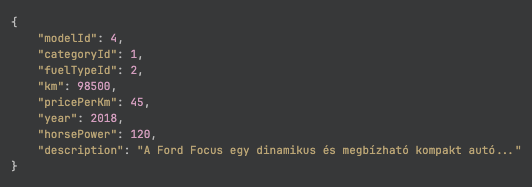
  
A válaszok JSON formátumban érkeznek, ebben az esetben az összes autót egy listában küldte vissza

A “GET /cars/1” az 1-es ID-jű autóval tér vissza  


Itt egyetlen autó objektum van a válaszban.

#### Feltöltés

A feltöltési kérések szintén nagyon hasonlóak. It a HTTP kérés body részében kell a feltöltendő adatnak lennie. A következőképpen néz ki egy feltöltés:

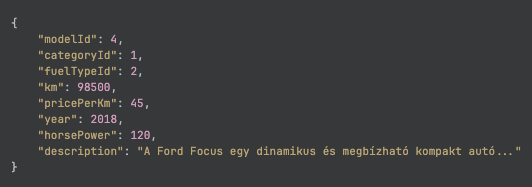
“POST /cars”  
A kérés body része:  
  
Amennyiben sikeres a feltöltés az API egy 200 OK kóddal tér vissza. Ellenkező esetben a hibának megfelelő kóddal tér vissza.

##### Képek

Képek feltöltése esetén ez némiképp máshogy működik. Mivel itt az autó id-jén kívül egy fájlt is meg kell adni, itt nem sima JSON, hamen Multipart Form formátumban kell az adatokat megadni. Ez sima HTTP lekérdezés formájában komplikált, ezért képek feltöltésére mindenképp az admin felületet használjuk, ahol ez el van rejtve.

#### 

#### Módosítás

A rekordok módosítása hasonlóképpen működik mint a feltöltésük, csak ebben az esetben a kérés része a rekord ID-je. Példa  
“PUT /cars/1”  
A kérés body része:  


Minden ugyanúgy zajlik mint a feltöltés esetében, csak itt a már meglévő rekord kerül frissítésre. Minden mezőt meg kell adni, ugyanis módosítás helyet csere történik. Sikeres csere esetén 204 No Content választ ad az API, ellenkező esetben a megfelelő hibakóddal tér vissza.

Képek esetén módosításra nincs lehetőség, ebben az esetben törölni kell a rekordot, majd egy újat létrehozni helyette.

#### Törlés

A törlések a legegyszerűbbek.  
“DELETE /cars/1”

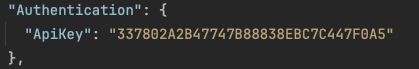
Semmi mást nem kell a lekérdezéshez megadni. Sikeres törlés esetén 204 No Content választ ad az API.

#### 

#### Autentikáció

Az API egy kulccsal, illetve CORS szolgáltatással van védve. Ezek POST, PUT és DELETE kérések esetén szükségesek, a GET kérések akárhonnan elérhetőek. Ezeket az API appsettings.json fájljában lehet módosítani.

##### API Key

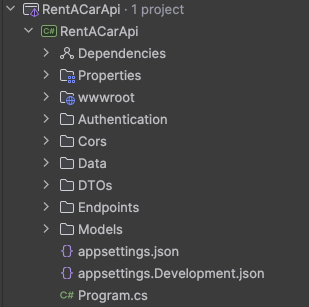
Az itt megadott kulcsot minden védett lekérdezés header részébe, x-api-key tulajdonság ként meg kell adni.  


##### CORS

A CORS az API böngészőben való használatakor csak a megengedett URL-ről érkező kéréseket engedi lekezelni. A védett kérések csak az admin felületünkről hajthatóak végre. Ennek az URL-ét szintén az appsettings.json fájlban lehet beállítani.  


### 

### 2.6 Program struktúrája

A program a következőképpen van elrendezve:  


* A Program.cs fájl tartalmazza a program kiindulási pontját. Itt történnek meg az alapvető beállítások és a szerver elindítása.
* Az Authentication mappa tartalmazza az autentikációval kapcsolatos segéd osztályokat.
* A Cors mappa tartalmazza a CORS beállítását segítő osztályokat.
* A Data mappa tartalmazza az adatbázissal kapcsolatos osztályokat, illetve az EF Core által generált migrációkat.
* A DTOs mappa tartalmazza a DTO osztályokat. Ezek egyszerű osztályok amelyek segítik az adatok küldését és feldolgozását.
* Az Endpoints mappa tartalmazza azokat a segéd osztályokat és metódusokat amelyek létrehozzák a végpontokat és lekezelik a rájuk érkező kéréseket.
* A Models mappa tartalmazza azokat az osztályokat, amelyek ténylegesen reprezentálják az adatbázisban lévő rekordokat.
* Az appsettings.json fájlok a beállítások egyszerű lekezelé teszik lehetővé.
* A wwwroot mappa statikus webbel kapcsolatos fájlokat tartalmazza, esetünkben ide kerülnek feltöltésre a képek.

### 2.7 API telepítése és futtatása

#### Szükséges csomagok

* Naprakész DOTNET SDK
* MariaDB 11.6.2 vagy újabb

#### Adatbázis előkészítése

A MariaDB adatbázis szervernek alapbeállítások szerint a 3306-os porton kell futnia. Kell rajta lenni egy “rent\_a\_car” adatbázisnak és egy “api\_user” felhasználónak, aminek “admin” a jelszava. Ezek elkészítéséhez a projekt mellett található egy SQL fájl, olyan néven hogy “rent\_a\_car\_database\_setup.sql”. Ezt a következőképpen lehet futtatni:  


Ez a script létrehozza a felhasználót és az adatbázist, ami azonban még üres. Ezt egy másik SQL fájl, a “rent\_a\_car\_dump.sql” fájl segítségével lehet feltölteni. Ezt a következőképpen kell futtatni:  


Ez létrehozta a táblákat és feltöltötte őket adattal.

#### API Futtatása

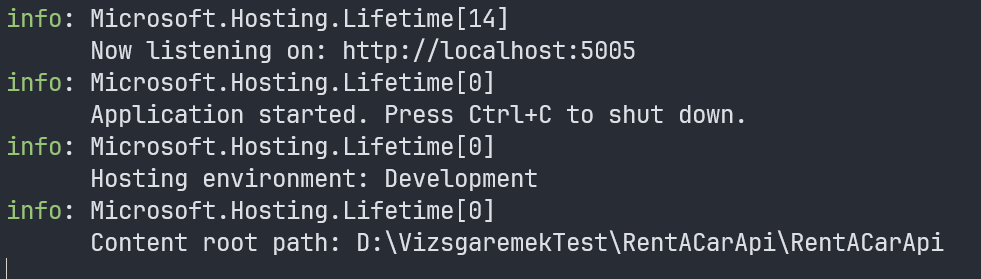
Az API futtatásához a “RentACarApi.csproj” fájl mellé kell belépni. Ez a “./RentACarApi/RentACarApi/” mappában található. Itt a következő dotnet parancsokat kell lefuttatni:



Ez a parancs letölti az összes szükséges NuGet csomagot. Ezután le lehet futtatni a projektet:



Ha minden sikeresen ment akkor hasonló szövegnek kell megjelennie a parancssorban:



Ekkor az API fut és készen áll a használatra. Alap beállítások szerint az 5005-ös portot fogja használni. A projekt másik 2 komponense erre a portra számít.

## 3. Frontend Weboldal

### 3.1 Bevezetés

Weboldalunk egy letisztult, egyszerűen használható keresési élményt nyújt felhasználóink számára. Rajta keresztül átnézhetik az elérhető járműveinket és egyszerűen felveheti a kapcsolatot ügyfélszolgálatunkkal.

### 3.2 Használt technológiák

#### HTML

HTML segítségével alakítottuk ki a weboldal struktúráját.

#### CSS + Bootstrap

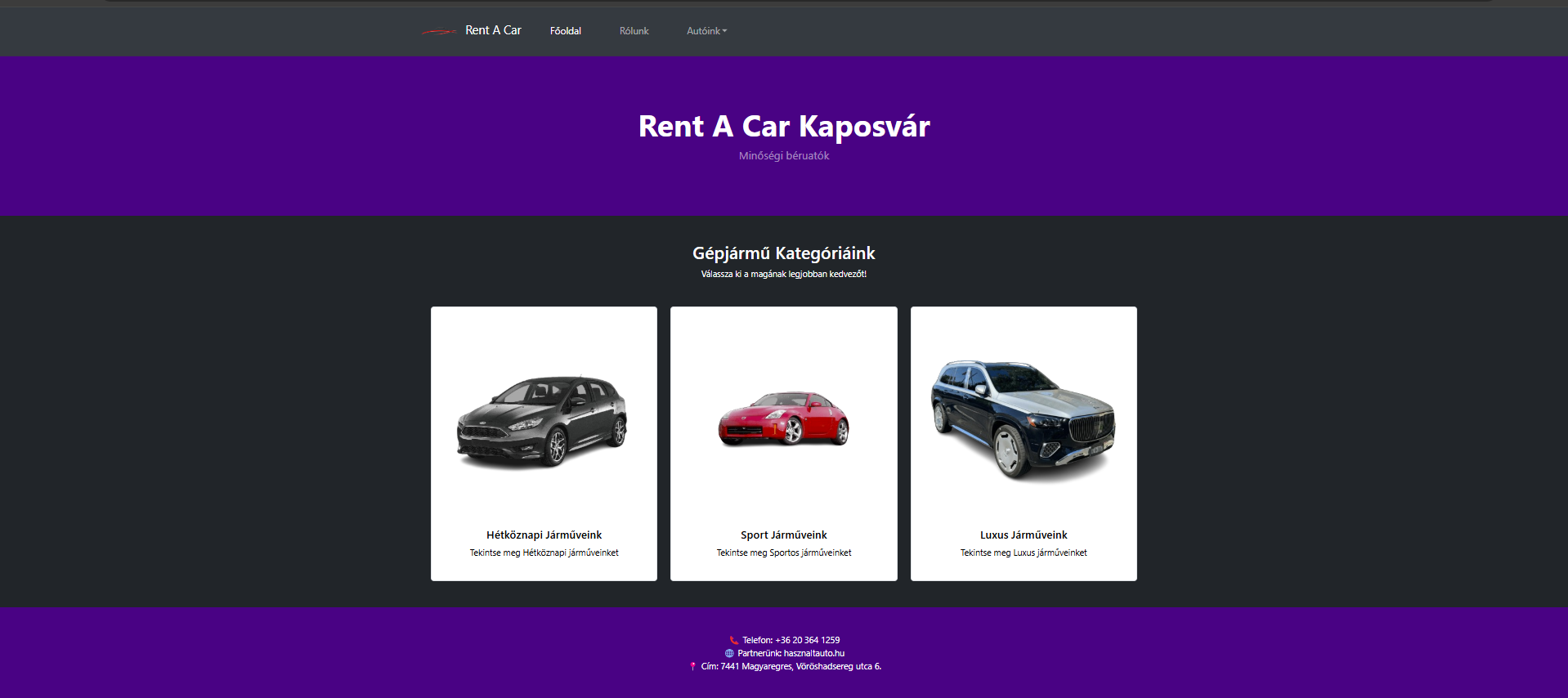
CSS és a Bootstrap keretrendszer segítségével alakítottuk ki az oldal stílusát, adtunk hozzá komplikáltabb elemeket és tettük reszponzívvá, hogy más méretű kijelzőn és mobil eszközökön keresztül is a megfelelő felhasználói élményt nyújtsuk.

#### Javascript

Javascript segítségével alakítottuk ki a weboldal dinamikus részeit. A fetch api használatával kapcsolódunk backend API-unkhoz, hogy lekérjük a benne lévő adatokat, illetve ezeket szűrők segítségével, rendszerezve jelenítsük meg a felhasználó számára.

### 3.3 Weboldal használata

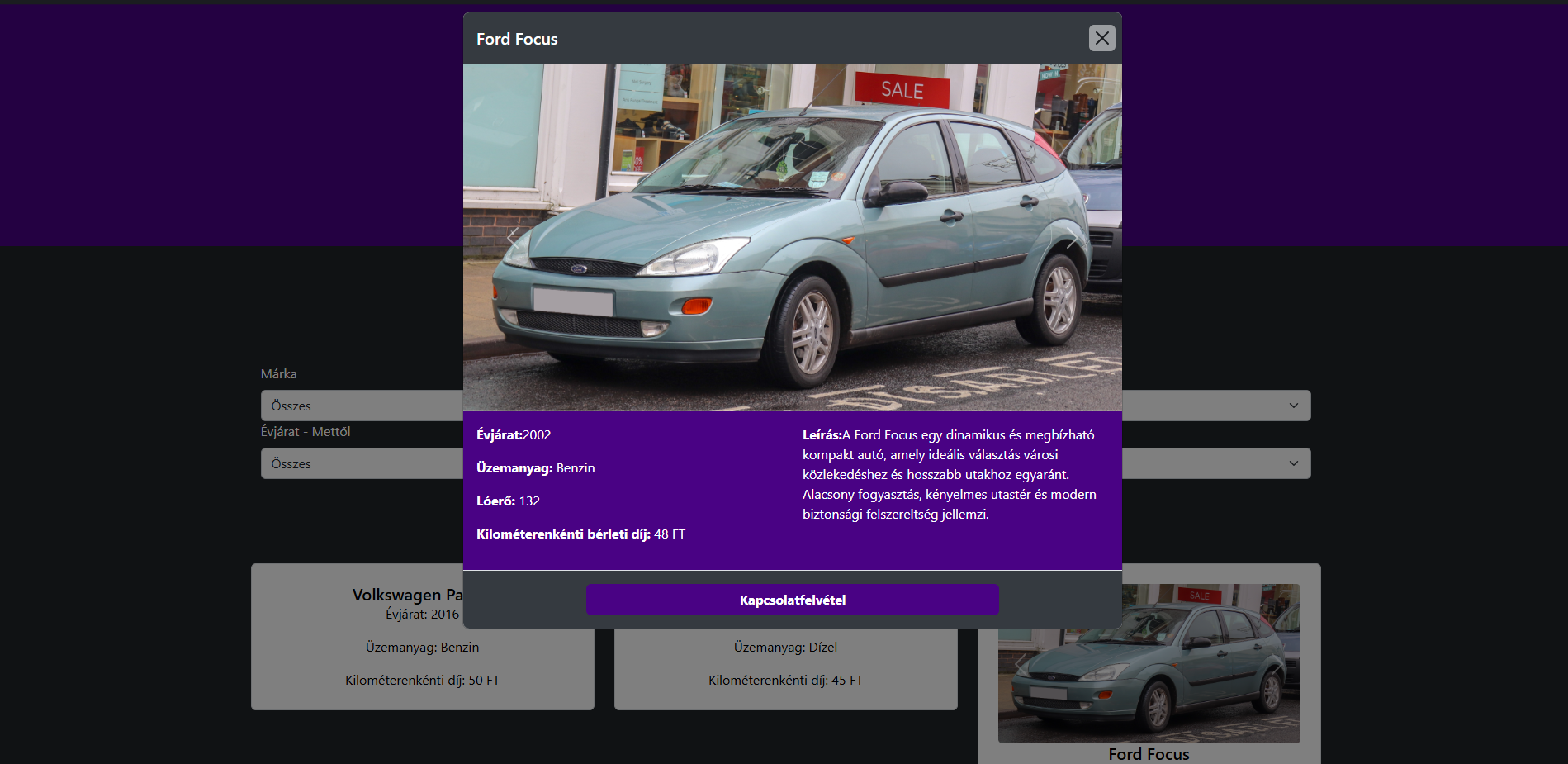
Az oldal megnyitásakor a főoldal jelenik meg, ahol a felhasználó szabadon böngészhet a menüpontok és a bérelhető autók kategóriái között: Hétköznapi, Sportos, Luxus.

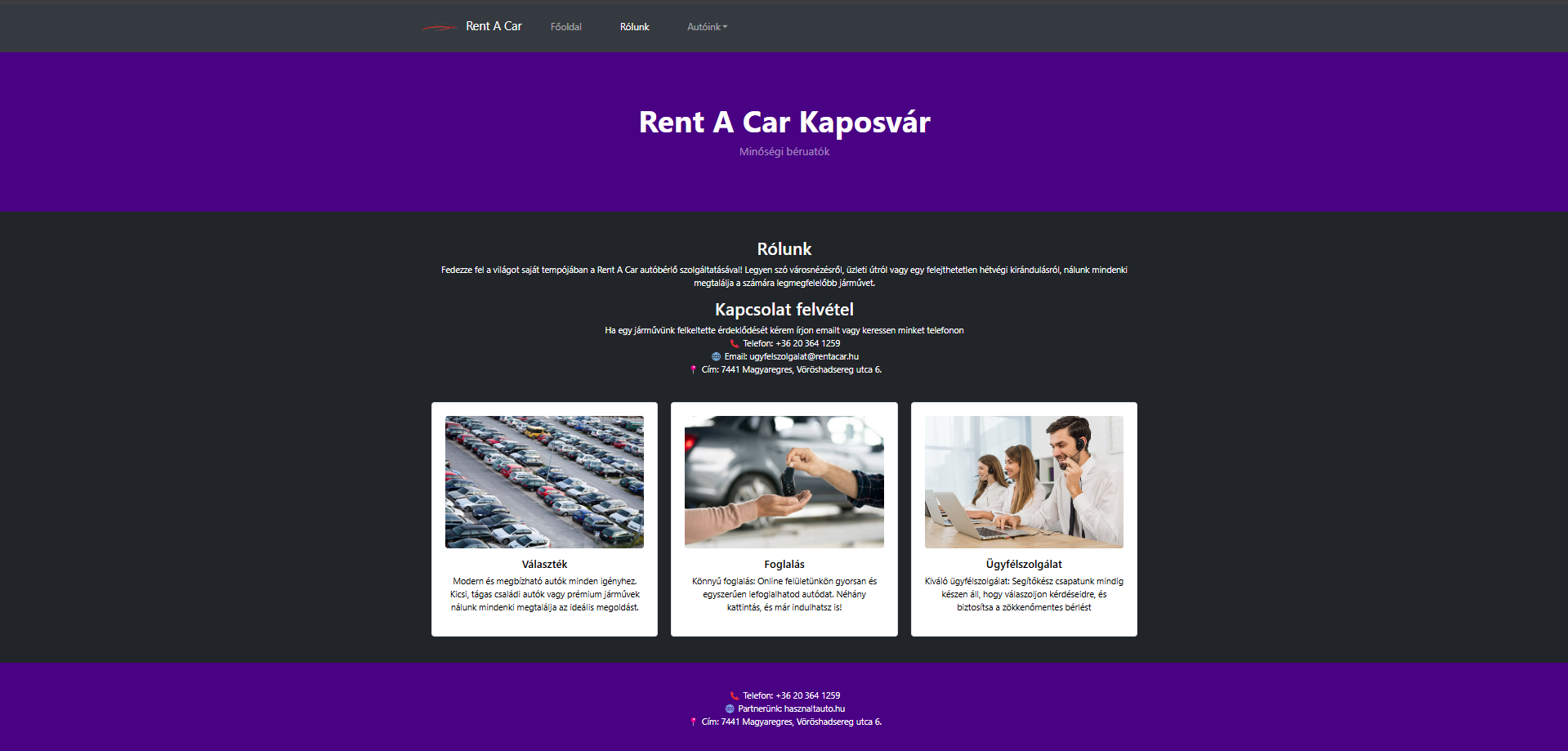


Kereső felület, lehetőséget ad szűrési feltételek segítségével kiválasztani az ideális autót a felhasználó igényei szerint.



A szűrési feltételeknek megfelelő gépjárművek a kereső ablak alatt jelennek meg majd a kiválasztott gépjárműre kattintva kapunk részeletes leírást.



Rólunk oldal, segítségével a felhasználó megismerkedhet a céggel és a weboldallal illetve lehetősége lesz a kapcsolatfelvételre is.  


### 3.4 Weboldal telepítése és futtatása

A weboldal teljesen statikus, minden a böngészőben történik. Amennyiben fut a megfelelő porton (5005) fut az API, minden további nélkül működnie kell. Az index.html, vagy akármelyik másik html fájl megnyitható böngészővel.

## 4. Adminisztrációs felület

### 4.1 Bevezetés

Adminisztrációs felületünk lehetőséget biztosít a szolgáltatás üzemeltetői számára, hogy egyszerű, interaktív és grafikus módon változtathassák az adatbázis és ezzel együtt a weboldal tartalmát is.

### 4.2 Használt Technológiák

#### HTML

Frontend, végfelhasználói oldalunkhoz hasonlóan HTML segítségével alakítottuk ki a weboldal struktúráját.

#### CSS

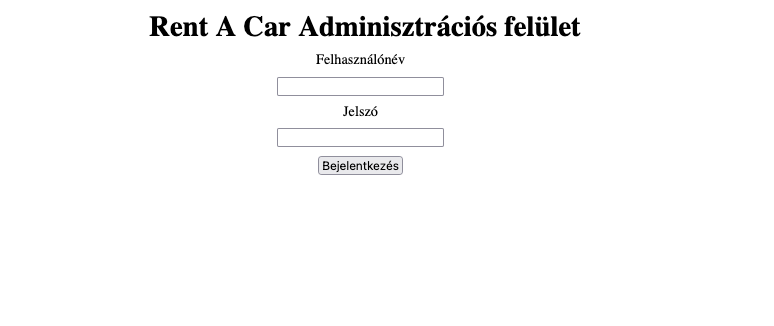
Ezen esetben kiegészítés nélküli, vanilla CSS-t használtunk, ugyanis ez a weboldal csak az üzemeltetők számára van tervezve, ahol a kinézet nem olyan fontos mint a kliensek esetében.

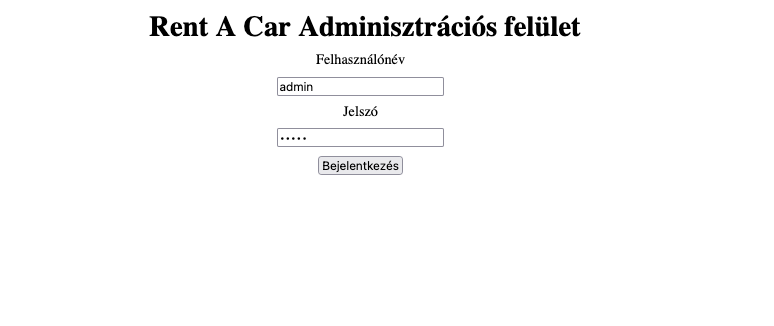
#### PHP

Ezúttal a nagyobb biztonság és dinamikusság kisebb fontossága miatt PHP-val van a felület logikájának nagy része elkészítve.

### 4.3 Adminisztrációs felület használata

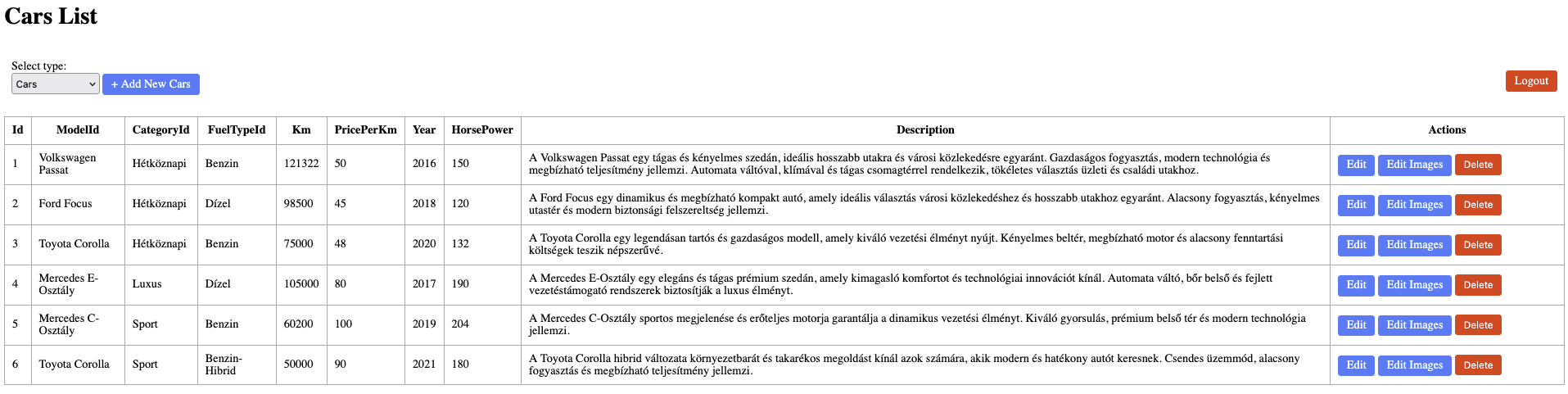
#### 4.3.1 Bejelentkezés

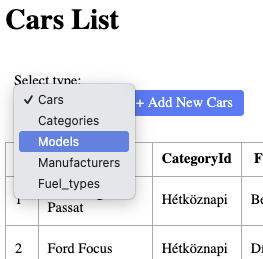
Amennyiben nem vagyunk bejelentkezve, a következő oldalra irányít minket a program:  


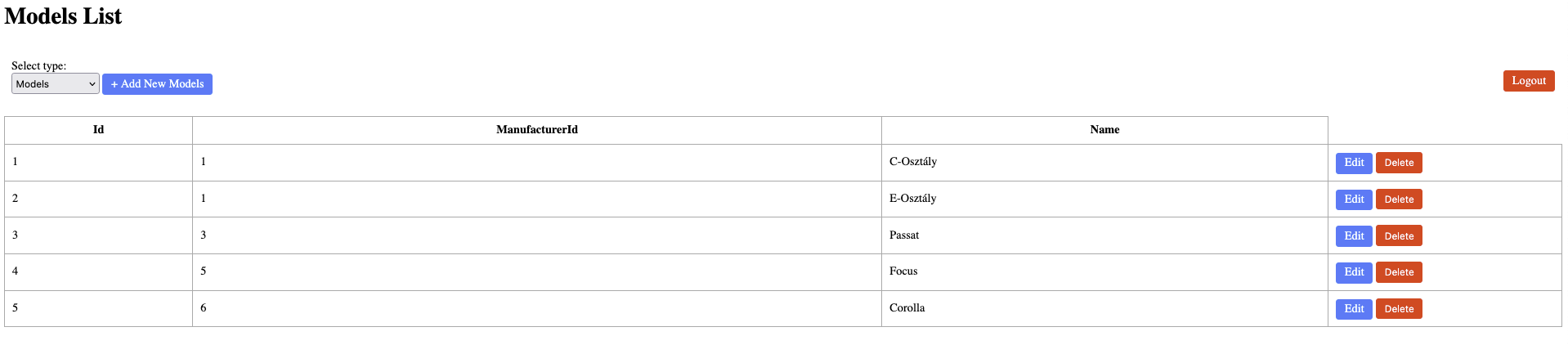
A szoftver egy egyszerű, lokális bejelentkezési rendszert alkalmaz. Egy fiókot tárol el a program, amit indításakor tudunk konfigurálni. Jelenlegi állapotban ez csak lokális használatra van tervezve, úgyhogy a védelem nem a legfontosabb feltétel. Ez később, bővülés esetén egy frissítésben változhat. Alapból a felhasználónév és a jelszó is az “admin” szóra van állítva. Ezeket a config.php fájlban lehet változtatni. Miután kitöltöttük mindkét mezőt nyomjunk a bejelentkezés gombra.  


Amennyiben hibásan jelentkeztünk be ezt a rendszer jelzi. Ellenkező esetben továbbléphetünk.

#### 4.3.2 Rekordok listázása

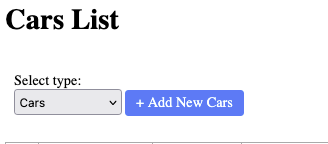
Miután sikeresen beléptünk a következő oldalon találjuk magunkat:  


Itt egy megadott típus összes rekordját láthatjuk. A típust a bal felső sarokban, legördülő menüvel választhatjuk ki.  


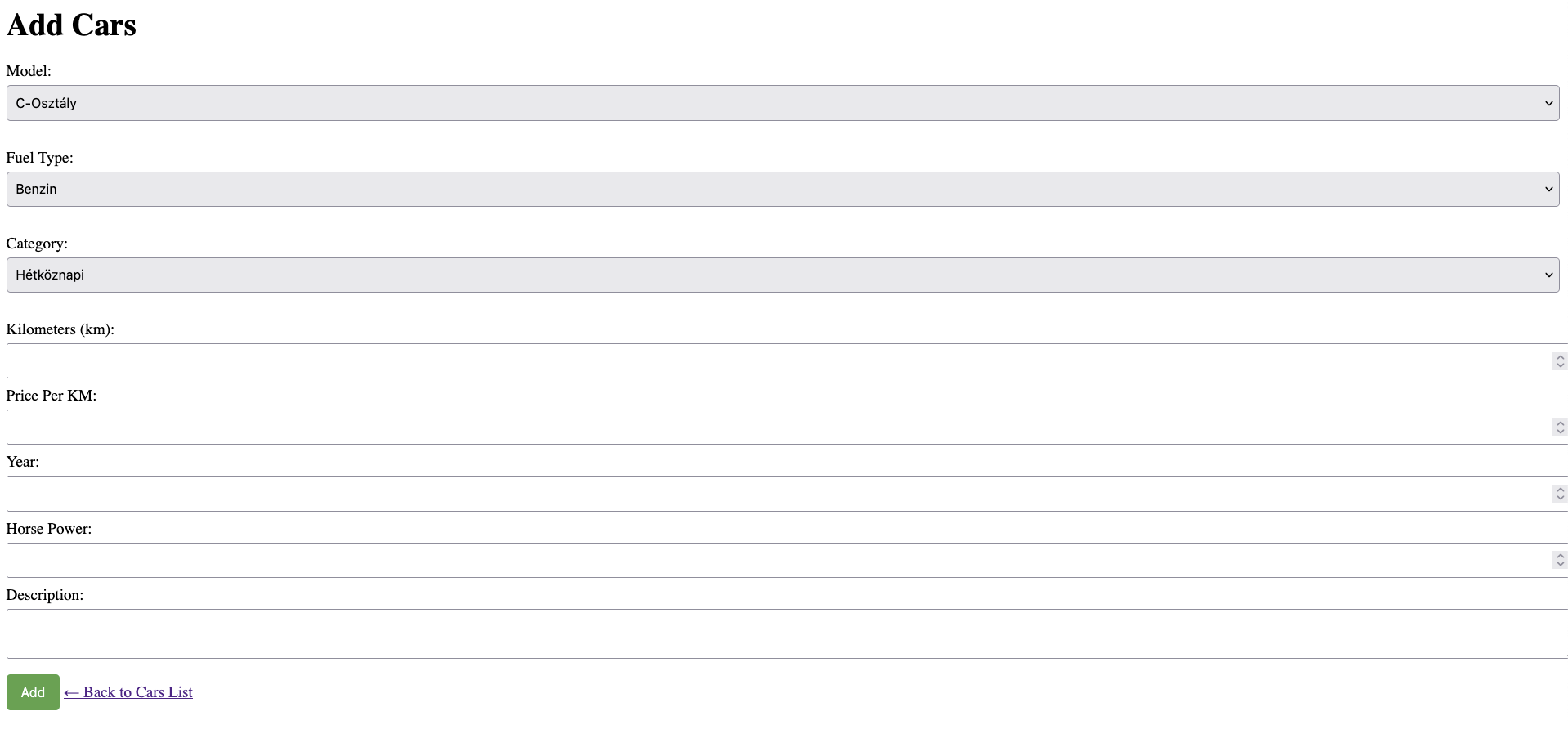
Miután ezt megtettük az oldal átvisz a megfelelő listára, ebben az esetben a modellekhez.  


#### 4.3.3 Új rekord hozzáadása

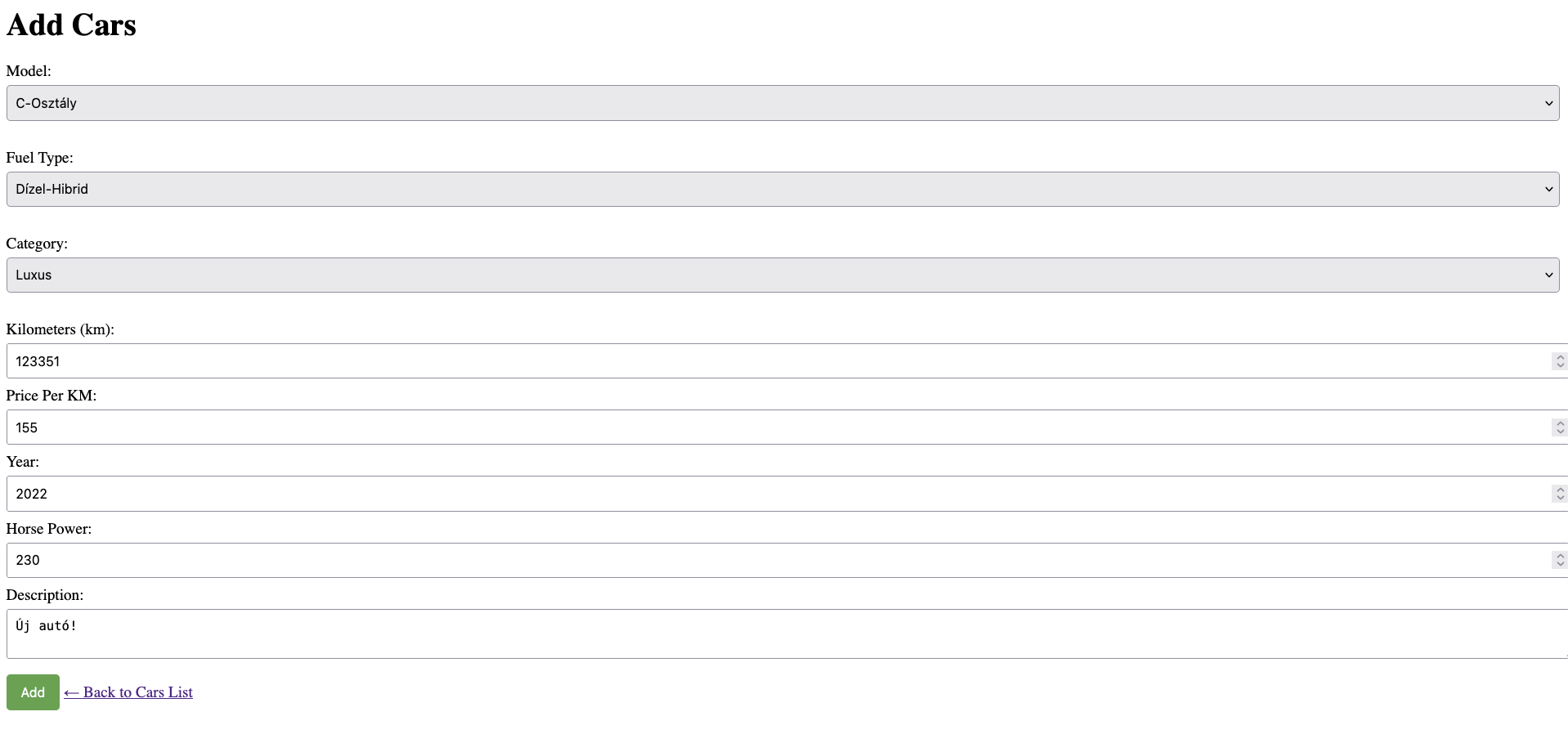
Új rekordot a típus választó menü melletti, “Add New” gombbal vehetünk fel.



Miután ezt megnyomtuk, a program egy másik oldalra visz át minket. Itt a választott típusnak megfelelően jelennek meg a beviteli mezők.



Miután kitöltöttük a mezőket, az “Add” gomb segitségével hozzáadjuk a rekordot. Amennyiben mégsem szeretnénk, a mellette lévő “Back to List” gombbal mentés nélkül visszaléphetünk.



Ezután láthatjuk a listán, hogy az új rekord fel lett véve.



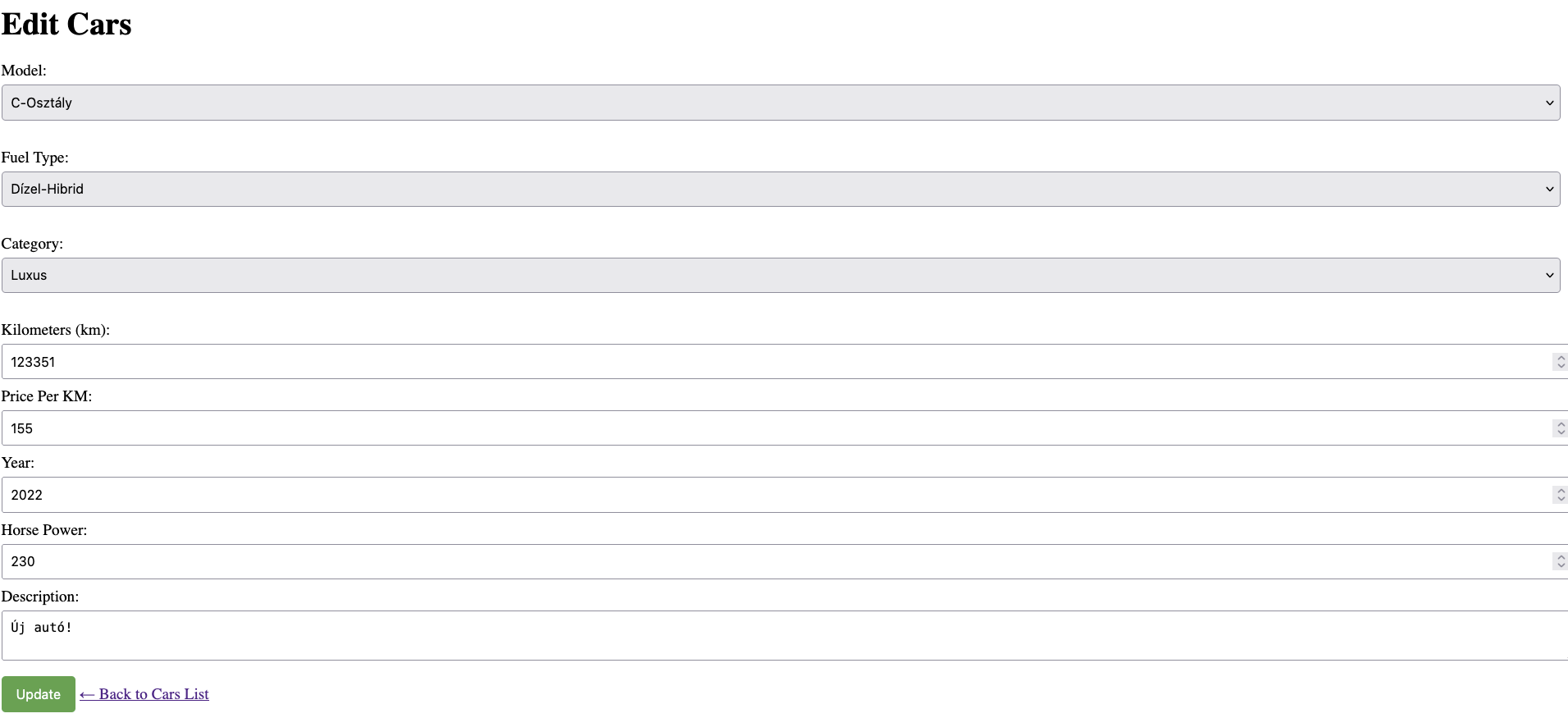
#### 

#### 4.3.4 Rekordok módosítása

Minden rekord mellett, az “Actions” oszlopon belül található egy “Edit” gomb.



Ha erre rányomunk, az előző felületre for elvinni, csak most már eleve ki vannak töltve a mezők.



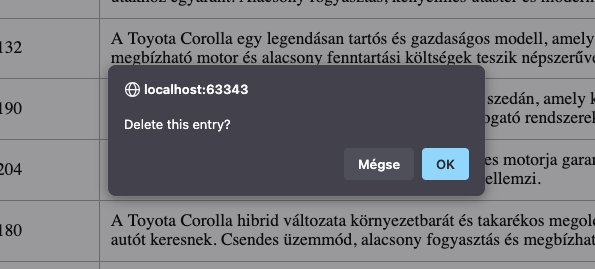
Miután átírtunk valamit, az “Update” gombra nyomással elmentjük a változások. A “Back to List” gombbal elvetjük a változásokat és visszalépünk a listára.



#### 

#### 4.3.4 Rekordok törlése

Az “Edit” gomb melletti, “Delete” gombbal törölhetőek ki a rekordok. Miután rányomunk, az oldal megerősítést kér. Amennyiben ezt elfogadjuk, a rekord törlésre kerül.



#### 

#### 4.3.5 Képek kezelése

Képeket csak az autókhoz lehet hozzáadni. Ezt egy külön menüpontban lehet megtenni, ahova az “Edit Images” gombbal lehet átlépni.

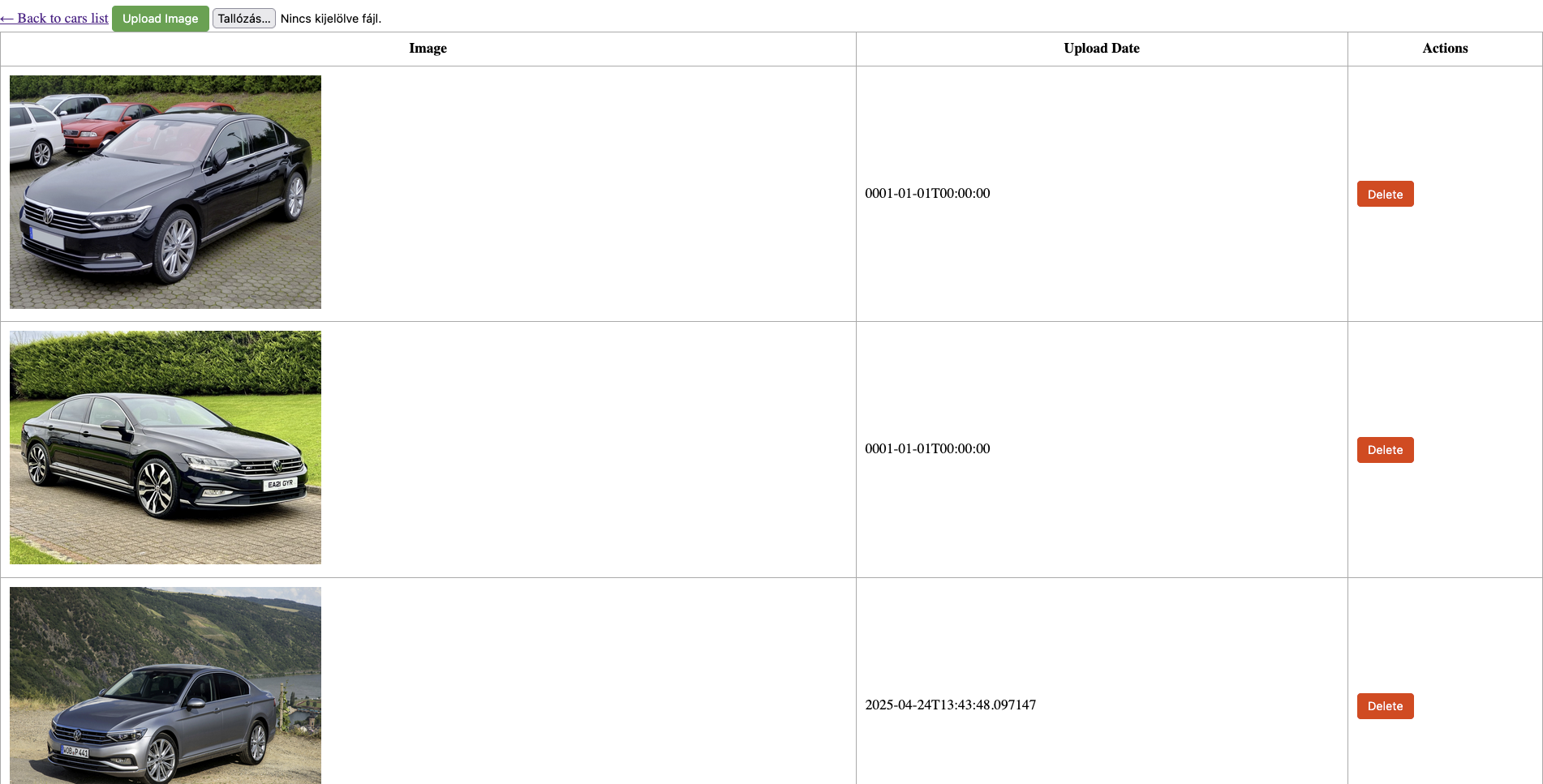


Amennyiben már vannak képen feltöltve az adott rekordhoz ezeked kilistázza.



Ellenkező esetben a lista üres lesz.

Itt az oldal tetején lévő tallózás gombbal ki tudunk választani egy képet a számítógépünkről. Ha ez megtörtént az oldal írni fogja a kép nevét.  


Ezután az “Upload Image” gombbal feltölthetjük a képet. Ekkor már láthatjuk is a lista alján.  


A képektől jobbra található “Delete” gombbal, a rekordokhoz hasonlóan törölhetjük őket.

#### 4.3.6 Kijelentkezés

A főoldalon lévő, jobb felső sarokban található “Logout” gomb megnyomásával kijelentkezhetünk a rendszerből.

### 4.4 Adminisztrációs felület telepítése és futtatása

#### Szükséges csomagok

* PHP 8.1 vagy újabb

#### Lokális PHP szerver futtatása

Amennyiben telepítve van a számítógépen a PHP, a következő paranccsal lehet elindítani egy lokális fejlesztői szervert. Ehhez az Index.php mellett kell lenni, ami a “./RentACarAdminPage/” mappában található.

Ez a “localhost:8081/” elindít egy szervert. Erre a linkre navigálva férhetünk hozzá az oldalhoz.

## 5. Jövőbeli fejlesztési lehetőségek

* Felhasználói regisztráció és bejelentkezés
* Foglalási rendszer
* Fizetési integráció
* Teljes konténerizálás Docker segítségével
* Cég növekedése esetén több adminisztrátori felhasználó támogatása

## 6. Tesztelési dokumentáció

A tesztelést manuálisan végeztük a teljes fejlesztési idő alatt. Első sorban az API-t teszteltük, erre REST klienseket, mit például Postman vagy Restfox, használtunk.

## 8. Fejlesztői csapat és munkamegosztás, csapatmunka

Horváth Dávid Alexander:

* Backend fejlesztés
* REST API megvalósítása
* Az adatbázis struktúrájának megtervezése

Gaál András Krisztián:

* Frontend fejlesztés
* Vizuális dizájn kialakítása
* Felhasználói élmény (UX) megtervezése és optimalizálása

**9. Rendszerkövetelmények**

### 9.1 Hardverkövetelmények

#### Minimális rendszerkövetelmények:

* **Processzor:** Intel Core i3 (vagy AMD ekvivalens)
* **Memória:** 4 GB RAM
* **Tárhely:** 10 GB szabad tárhely
* **Kijelző:** 1280x720 felbontás
* **Hálózat:** Stabil internetkapcsolat

#### Ajánlott rendszerkövetelmények:

* **Processzor:** Intel Core i5 vagy újabb (vagy AMD Ryzen 5 ekvivalens)
* **Memória:** 8 GB RAM vagy több
* **Tárhely:** SSD, minimum 20 GB szabad tárhellyel
* **Kijelző:** Full HD (1920x1080) vagy jobb
* **Hálózat:** Stabil, nagysebességű internetkapcsolat

### 9.2 Szoftverkövetelmények

* **Böngésző:** Friss böngésző
* **Operácíós rendszer:** Naprakész Windows, Linux vagy MacOS verzíó

#### Fejlesztői környezet:

* **Operációs rendszer:** Windows 10/11, Linux (Ubuntu 20.04+), macOS
* **Fejlesztői eszközök:**
  + **Visual Studio 2022** (ajánlott az ASP.NET Core fejlesztéshez)
  + **Visual Studio Code** (ajánlott frontend fejlesztéshez)
* **Egyéb szoftverek:**
  + **.NET SDK 6.0+**
  + **Node.js** (ajánlott frontend build rendszerekhez)
  + **Docker** (opcionális konténerizációhoz)
  + **MariaDB vagy MySQL** (adatbázis kezelő)