

**Budapesti Műszaki Szakképzési Centrum**  
**Verebély László Techinkum**

**Car-RentAll**

Készítették: G. Szabó Zita

Nagy Ferenc

Tóth Ádám

Konzulens: Juhász Zoltán

2022

# NYILATKOZAT A VIZSGAREMEK ÉS A DOKUMENTÁCIÓ EREDETISÉGÉRŐL

Alulírott G. Szabó Zita, Nagy Ferenc és Tóth Ádám a **BMSZC Verebély László Technikum 5-0613-12-03** azonosító számú **Szoftverfejlesztő és -tesztelő** képzésében részt vevő hallgatói büntetőjogi felelősségünk tudatában nyilatkozunk és aláírásunkkal igazoljuk, hogy a ***Car-RentAll*** című vizsgaremek és dokumentáció saját, önálló munkánk, és abban betartottuk az iskola által előírt, a vizsgaremek készítésére vonatkozó szabályokat.

Tudomásul vesszük, hogy a dokumentációban plágiumnak számít:

- szó szerinti idézet közlése idézőjel és hivatkozás nélkül,
- tartalmi idézet hivatkozás megjelölése nélkül,
- más publikált gondolatainak saját gondolatként való feltüntetése.

E nyilatkozat aláírásával tudomásul vesszük továbbá, hogy plágium esetén vizsgaremekünk visszautasításra kerül.

Budapest, 2022.05.03

G. Szabó Zita

Nagy Ferenc

Tóth Ádám

## Tartalom

1. Bevezetés.....	4
2. Fejlesztői dokumentáció.....	5
2.1. Fejlesztői környezet és alkalmazott technológiák.....	5
2.2. Ütemterv.....	6
3. Az alkalmazás inicializálása.....	9
3.1 Az MVC Architektúra.....	9
3.2 Project létrehozása.....	10
3.3 Modellek.....	12
3.4 Az Entity Framework.....	15
3.5 Vezérlők.....	17
3.6 Nézetek.....	22
3.7 API és a RESTful konvenció.....	23
3.8 DTO és az AutoMapper.....	24
3.9 API-k az alkalmazásban.....	24
3.10 Bejelentkezés és hitelesítés.....	27
3.11 OAuth.....	27
4. Az asztali alkalmazás.....	28
4.1 Adatbázis.....	28
4.2 Bejelentkezés.....	30
4.3 Navigáció.....	32
4.4 Kereső és szűrő feltételek.....	33
4.5 Új adat hozzáadása.....	34
4.6 Adat törlése.....	34
5. Felhasználói dokumentáció.....	35
5.1 Webalkalmazás.....	35

5.2 Asztali alkalmazás.....	42
6. Összegzés .....	50
6.1 Tesztesetek .....	50
6.2 Továbbfejlesztési lehetőségek.....	52
6.3 Összegzés .....	53
Források.....	54
Ábrajegyzék.....	55

# 1. Bevezetés

Autókölcsönzéssel kapcsolatos webalkalmazás, mobilapplikáció ma már számtalan van. Rengeteg helyen megtalálhatjuk a számunkra legideálisabb járművet, melyet néhány napra, vagy akár hosszú időre is ki tudunk bérelni. Hatalmas a választék nemcsak a jármű tulajdonságai, de a kölcsönzés ára alapján is. A kölcsönzés mellé általában több extrát is biztosítanak a bérbeadással foglalkozó cégek: biztosítást, klubtagságot, mely különböző mértékű kedvezményekkel jár stb. Több oldalon tekinthetünk meg értékeléseket az előző ügyfelektől.

Célunk egy olyan kombinált alkalmazás fejlesztése volt, mely rendelkezik a felhasználó számára elérhető online felülettel és a telephelyen használható belső felülettel is. Az online felületen admin jogosultsággal bíró felhasználó a kölcsönzési igényeket, az ügyféladatokat és a járműadatokat kezelni tudja, míg a felhasználó saját személyes adatokat adhat hozzá regisztráció után, módosíthatja azokat és járműveket bérelhet. A saját jóváhagyásra váró szerződéseit, folyamatban lévő szerződéseit és lejárt szerződéseit megtekintheti, visszavonhatja igényét.

Fontos szempont volt, hogy a webes felület könnyen átlátható és kezelhető legyen, a felhasználó ne vesszen el a rengeteg információ között és ne érezze úgy, hogy csak rá akarjuk erőszakolni a járműveinket. A böngészés gyors, egyszerű és kényelmes legyen.

A telephelyi alkalmazásban vezetjük a jármű általános adatai mellett a műszaki állapotára vonatkozó adatokat, a kiadáskori és visszavételi kilométeróra állásokat, a jármű felszereltségét, az esetleges sérüléseket kiadáskor és átvételkor, a benzinmennyiséget stb. Lehetőségünk van a munkahelyen dolgozó személyzet adatainak, jogosultságainak a kezelésére, járműkiadásra, szerződéskötésre és a járművek nyomon követésére. Új ügyfelet adhatunk hozzá, szerkeszthetjük a meglévő ügyfelek tulajdonságait, vagy törölhetjük azokat. Ez vonatkozik a járművekre is.

A vizsgaremeket két részre szedtük. Az első részt ASP.NET MVC segítségével valósítottuk meg, a második részt Windows Forms segítségével.

A dokumentáció első részében az ASP.NET MVC részt fogjuk ismertetni.

## **2. Fejlesztői dokumentáció**

### **2.1. Fejlesztői környezet és alkalmazott technológiák**

#### **Visual Studio 2019 és 2022**

A Visual Studio 2022 a Microsoft integrált fejlesztői környezetének legújabb változata, melyet fejlesztők tömegei használnak világszerte. A Visual Studio 2022 segítségével felhasználóbarát módon hozhatunk létre számítógépes programokat és mobilalkalmazásokat, továbbá többféle weboldalt, webes alkalmazást, vagy bármilyen egyéb webes szolgáltatást.

A Visual Studio programcsalád hagyományosan a Microsoft fontosabb szoftverfejlesztési platformjait egyesíti, ezáltal egy környezetben használhatjuk a cég fontos szoftverfejlesztési eszközeit. A Visual Studio 2022 például kiválóan használható C#, C, C++, Python és Java programnyelvek kódolásához.

A Visual Studio 2022 számos előremutató fejlesztéssel érkezik, emellett az új kiadás egyik legfontosabb fejlesztése a Visual Studio 2019 egyik visszatérő panaszát is megoldja, nevezetesen az IDE rendszer intenzív memóriaigényét. A nagy számítási teljesítményt igénylő, összetett alkalmazásokon dolgozó fejlesztők számára ez óriási előnyt jelenthet.

Sokat javítottok a kódkiegészítő funkcionalitásán, a Visual Studio 2022 intelligens kód funkciója a kódolási kontextus jobb megértése révén akár egész kódsorokat képes magától kitölteni.

A Visual Studio 2022 felülete is változott: elődeinél felhasználóbarátabb és intuitívabb kódolási környezet felületet kínál. Ez magában foglalja a szoftverkörnyezet megjelenésének testreszabási lehetőségeinek bővítését (például a Windows témájához igazítva), valamint a felhasználó számára optimálisabb dokumentumkezelő rendszer telepítésének lehetőségét. [1]

#### **C# programozási nyelv**

A C# a Microsoft által a .NET keretrendszer részeként kifejlesztett objektumorientált programozási nyelv. A nyelv alapjául a C++ és a Java szolgált.

A C# az a programozási nyelv, ami a legközvetlenebb módon tükrözi az alatta működő, minden .NET programot futtató .NET keretrendszert, valamint erősen függ is attól: nincsen nem menedzsel, natív módban futó C# program. A primitív adattípusai objektumok, a .NET típusok megfelelői. Szemétgyűjtést használ, valamint az absztrakcióinak többsége (osztályok, interfészek, delegáltak, kivételek) a .NET futtatórendszert használja közvetlen módon.

A legtöbb programozási nyelvtől eltérően a C# megvalósítások nem rendelkeznek önálló, eltérő osztály- vagy függvénykönyvtárakkal. Ehelyett a C# szorosan kötődik a .NET keretrendszerhez, amitől a C# kapja a futtató osztályait és függvényeit. A .NET keretrendszer osztálykönyvtárát tartalmaz, ami a .NET nyelvekből felhasználható egyszerű feladatok (adatrepresentáció és szövegmanipuláció) végrehajtásától kezdve a bonyolult (dinamikus ASP.NET weblapok generálása, XML feldolgozás és reflexió) feladatokig. A kód névterekbe van rendezve, mely a hasonló funkciót ellátó osztályokat fogja össze. [2]

## **Azure**

Az Azure Cloud platform több mint 200 terméket és felhőalapú szolgáltatást foglal magába, amelynek célja, hogy segítsenek az új megoldások létrehozásában, melyekkel megfelellhet a mai kihívásoknak, és felkészülhet a jövőre. Alkalmazásokat fejleszthet, helyezhet üzembe és kezelhet akár a felhőben, akár a helyszínen vagy a peremhálózaton, és ehhez szabadon megválaszthatja saját eszközeit és keretrendszerét. [6]

## **abplusz**

Az abplusz egy webtárhely szolgáltató. A webtárhely szolgáltatás keretein belül több domaint is el lehet helyezni. [7]

## **2.2. Ütemterv**

### **2.2.1. Online felület**

<b>Feladat megnevezése</b>	<b>Időszak kezdete</b>	<b>Időszak vége</b>
A szükséges modellek és vezérlők megtervezése és létrehozása	2022.03.12	2022.03.15

Action és Nézet tervezés, létrehozás	2022.03.15	2022.03.18
Adatbázis és táblák létrehozása Code First migrációkkal	2022.03.18	2022.03.20
Adatbázis seedelés	2022.03.20	2022.03.24
Form hozzáadás	2022.03.25	2022.03.29
Form címkék, legördülő listák megtervezése és hozzáadása	2022.03.29	2022.04.02
Adat mentés és szerkesztés lehetőségeinek hozzáadása	2022.04.02	2022.04.06
Validáció hozzáadása, egyedi validációk létrehozása	2022.04.06	2022.04.08
Kliens oldali validáció és Anti-forgery tokenek hozzáadása	2022.04.08	2022.04.08
Web API-k tervezése, implementálása és tesztelése	2022.04.08	2022.04.12
Data Transfer Objectek hozzáadása	2022.04.12	2022.04.14
Auto Mapper használata	2022.04.14	2022.04.15
Adattáblák beépülő modul hozzáadása és használata JQuery-vel	2022.04.15	2022.04.21
Bejelentkezés és hitelesítés implementálása	2022.04.24	2022.04.26
Felhasználók és jogosultságok hozzáadása	2022.04.26	2022.04.28
OAuth. Facebook és Google bejelentkezési lehetőségek hozzáadása	2022.04.28	2022.05.02
Teljesítmény optimalizáció	2022.05.02	2022.05.03

### 2.2.2. Asztali alkalmazás

Feladat megnevezése	Időszak kezdete	Időszak vége
Projekt cél- és feladatrendszerének	2022.01.03.	2022.01.05.



meghatározása		
Lokális adatbázis tervezése, létrehozása	2022.01.03.	2022.01.18.
Alap projekt felépítése a WFA-ban	2022.01.05.	2022.01.20.
Bejelentkezés és kilépés megvalósítása	2022.01.19.	2022.01.22.
Main form designjának tervezése, kivitelezése	2022.01.19.	2022.02.28.
Szerződések beolvasása, megjelenítése	2022.01.20.	2022.01.28.
Online adatbázis létrehozása, alkalmazása	2022.01.21.	2022.02.18.
Szűrő, kereső feltételek	2022.01.27.	2022.02.03.
Új szerződés form létrehozása	2022.01.30.	2022.02.06.
Szerződések lezárása menüpont	2022.02.05.	2022.02.10.
Gépjárművek listája form létrehozása, gépjárművek megjelenítése DGV-ban	2022.02.08.	2022.02.16.
Gépjárművek szűrési feltételeinek és Gépjármű törlése gomb létrehozása	2022.02.13.	2022.02.24.
Új gépjármű és Gépjármű adatainak szerkesztése menüpont létrehozása	2022.02.19.	2022.03.08.
Main form újra desingnálása panelek segítségével	2022.02.20.	2022.03.02.
Felhasználók form létrehozása	2022.03.03.	2022.03.07.
Felhasználók form gombjainak kivitelezése	2022.03.05.	2022.03.15.
Beállítások menüpont létrehozása	2022.03.08.	2022.03.14.
Gépjárművek listája pictureBox	2022.03.15.	2022.03.23.
Kinézet újragondolása	2022.03.10.	2022.03.20.
Sötétmód és színváltás	2022.03.10.	2022.04.20.
Teljesítmény optimalizáció	2022.04.25.	2022.05.03.

## 3. Az alkalmazás inicializálása

### 3.1 Az MVC Architektúra

Az MVC jelentése: Model-View-Controller

Magyarul: Modell-Nézet-Vezérlő

Egy ASP.NET MVC webalkalmazásban a fenti résztvevők, mint építőkövek az alábbi feladatokat látják el:

#### Modell

Az alkalmazás által kezelt információk tartomány-specifikus ábrázolása. A tartománylogika jelentést ad a puszta adatnak.

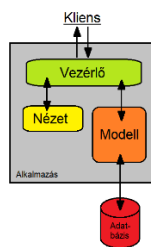
Sok alkalmazás használ állandó tároló eljárásokat (mint mondjuk egy adatbázis) adatok tárolásához. Az MVC nem említi külön az adatelérési réteget, mert ezt beleérti a modellbe. [3]

#### Nézet

Megjeleníti a modellt egy megfelelő alakban, mely alkalmas a felhasználói interakcióra, jellemzően egy felhasználói felületi elem képében. Különböző célokra különböző nézetek létezhetnek ugyanahhoz a modellhez. [3]

#### Vezérlő

Az eseményeket, jellemzően felhasználói műveleteket dolgozza fel és válaszol rájuk, illetve a modellben történő változásokat is kiválthat. [3]



1. ábra – MVN Reprezentáció

## 3.2 Project létrehozása

Template:

- ASP.NET Web Application (.NET Framework)

Framework:

- .NET Framework 4.8

A részletes beállítási lehetőségeknél az alábbiakat választottuk ki:

- Template: MVC
- Authentication: Individual Accounts

A project létrejötte után első dolgunk a NuGet Packages frissítése (Project -> Manage NuGet Packages ->Updates), a BundleConfig átalakítása (eltávolítjuk a „Script” részt minden Bundle elől) és a navigációs menü (NavBar) kijavítása volt, mivel az alap menü nem kompatibilis a Bootstrap használt, 5.1.3-as verziójával.

A következő lépés a \_NavBar Partial View létrehozása és a \_Layout.cshtml átalakítása.

A \_NavBar az alábbiak szerint épül fel a projectben

```
<nav class="navbar navbar-expand-lg navbar-dark bg-primary fixed-top">
  <div class="container-fluid">
    <a class="navbar-brand">
      @Html.ActionLink("Car-RentAll", "Index", "Home", new { area = "" }, new { @class = "navbar-brand" })
    </a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarResponsive"
      aria-controls="navbarResponsive" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarResponsive">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          @Html.ActionLink("Szerződések", "Index", "Szerzodesek", null, new { @class = "nav-link" })
        </li>
        <li class="nav-item">
          @Html.ActionLink("Ügyfelek", "Index", "Ugyfelek", null, new { @class = "nav-link" })
        </li>
        <li class="nav-item">
          @Html.ActionLink("Járművek", "Index", "Jarmuvek", null, new { @class = "nav-link" })
        </li>
      </ul>
      @Html.Partial("_LoginPartial")
    </div>
  </div>
</nav>
```

2. ábra: \_NavBar Partial View (Admin)

```

<nav class="navbar navbar-expand-lg navbar-dark bg-primary fixed-top">
  <div class="container-fluid">
    <a class="navbar-brand">
      @Html.ActionLink("Car-RentAll", "Index", "Home", new { area = "" }, new { @class = "navbar-brand" })
    </a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarResponsive"
      aria-controls="navbarResponsive" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarResponsive">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          @Html.ActionLink("Szerződéseim", "Index", "Szerzodesek", null, new { @class = "nav-link" })
        </li>
        <li class="nav-item">
          @Html.ActionLink("Saját adatok", "Index", "Ugyfelek", null, new { @class = "nav-link" })
        </li>
        <li class="nav-item">
          @Html.ActionLink("Bérelhető járművek", "Index", "Jarmuvek", null, new { @class = "nav-link" })
        </li>
      </ul>
      @Html.Partial("_LoginPartial")
    </div>
  </div>
</nav>

```

3. ábra: \_NavBar Partial View (Ügyfél)

Mivel az ügyfeleknek és az admin jogosultsággal rendelkező felhasználóknak különböző jogosultságaik vannak, emiatt szükséges volt két különböző navigációs menüt létrehozni. Míg az ügyfelek a saját szerződéseit, a saját személyes adatait és a bérelhető járműveket tudják megtekinteni, addig az admin jogosultsággal rendelkező felhasználó hozzáfér az összes elfogadásra váró, elfogadott és archivált szerződéshez, minden ügyfél adataihoz és a járművek adataihoz is. Lehetősége van új ügyfeleket, járműveket, szerződéseket hozzáadni és a meglévőket módosítani.

A \_Layout a navigációs menü külön nézetbe való áthelyezésével az alábbiak szerint alakul át:

```

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>@ViewBag.Title - Car-RentAll</title>
  @Styles.Render("~/Content/css")
  @Scripts.Render("~/bundles/modernizr")
</head>
<body>
  @if (!User.IsInRole("CanManage"))
  {
    @Html.Partial("_NavBarUgyfel");
  }
  else
  {
    @Html.Partial("_NavBar");
  }
  <div class="container body-content">
    @RenderBody()
    <hr />
    <footer>
      <p>&copy; @DateTime.Now.Year - Car Rent-All</p>
    </footer>
  </div>
  @Scripts.Render("~/bundles/Lib")
  @RenderSection("scripts", required: false)
</body>
</html>

```

4. ábra: \_Layout

A fenti ábrából kitűnik, hogy a megjelenített navigációs menüt a bejelentkezett felhasználó jogosultsága határozza meg. Amennyiben az oldalt nem bejelentkezett felhasználó használja, a 3. ábrán látható navigációs menü jelenik meg (Ügyfél).

### 3.3 Modellek

Az alkalmazásban több modellt megkülönböztetünk. Ezek közül a legfontosabbak:

- Ügyfél modell
- Jármű modell
- Szerződések modell
- Archívum modell

```
namespace Car_Rent_All.Models
{
    [Table("Ügyfelek")]
    public class Ugyfel
    {
        [Key, Required]
        public int Id { get; set; }

        [DisplayName = "Név:")]
        [Required(ErrorMessage = "A név megadása kötelező!"), MaxLength(60), Column(TypeName = "VARCHAR")]
        public string Nev { get; set; }

        [DisplayName = "Cím:")]
        [Required(ErrorMessage = "A cím megadása kötelező!"), MaxLength(200), Column(TypeName = "VARCHAR")]
        public string Cim { get; set; }

        [DisplayName = "Születési dátum:")]
        public DateTime? SzuletesiDatum { get; set; }

        [DisplayName = "Jogosítvány azonosító:")]
        [Required(ErrorMessage = "Az azonosító okmány számának megadása kötelező!")]
        public string Jogositvany { get; set; }

        [DisplayName = "Telefonszám:")]
        [Phone(ErrorMessage = "A megadott telefonszám nem egy érvényes telefonszám!")]
        [Required(ErrorMessage = "A telefonszám megadása kötelező!"), Column(TypeName = "VARCHAR")]
        public string Telefonszam { get; set; }

        [DisplayName = "E-mail cím:")]
        [EmailAddress(ErrorMessage = "A megadott e-mail cím nem egy érvényes e-mail cím!")]
        [Required(ErrorMessage = "Az e-mail cím megadása kötelező!"), Column(TypeName = "VARCHAR")]
        public string Email { get; set; }
    }
}
```

#### 4. ábra: Ügyfél modell

Az ügyfél modell tartalmazza az ügyfél személyes adatait (név, cím, születési dátum), a jogosítvány azonosítóját és az elérhetőségeit (telefonszám, e-mail cím). Mint az az 5. ábrán látszik, a modell esetében több annotációt használunk. Ezek közül szeretnénk a legfontosabbakat kiemelni:

- Table[„Táblanév”] – ez az annotáció garantálja azt, hogy a modell megírása után, a Code First migrációval létrejövő adatbázis táblának az idézőjelek közt megadott név legyen az elnevezése. Alapesetben, amennyiben nem használunk ehhez külön annotációt, a létrejövő tábla neve a modell megnevezése többes számban angolul. A mi esetünkben „Ugyfels”.

- [Key] – ez az annotáció garantálja, hogy a vele megjelölt tulajdonság elsődleges kulcs legyen az adatbázis táblában.
- [Required] – az annotációval megjelölt tulajdonságok kötelező elemek. Mint az ábrán látjuk több esetben specializáljuk a hibaüzenetet, amit akkor kapunk, ha a kötelező mező nem kerül kitöltésre.
- [Display(„Megnevezés”)] – az idézőjelek közt megadott elnevezés lesz a tulajdonság kiírt neve ott, ahol az megjelenik.

A „?” jellel megjelölt tulajdonság nem kötelező. A fenti ábrán a születési dátum megadása nem kötelező elem.

A Jármű modell az alábbiak szerint épül fel:

```
[Table("Jarmuvek")]
public class Jarmu
{
    [Key, Required]
    public int Id { get; set; }

    [Display(Name = "A jármű neve:")]
    [Required(ErrorMessage = "A jármű nevének megadása kötelező!")]
    public string Nev { get; set; }

    [Display(Name = "A jármű rendszáma:")]
    [Required(ErrorMessage = "A jármű rendszámának megadása kötelező!")]
    public string Rendszam { get; set; }

    [Display(Name = "A jármű alvázszáma:")]
    [Required(ErrorMessage = "A jármű alvázszámának megadása kötelező!")]
    public string Alvazszam { get; set; }

    [Display(Name = "A jármű váltójának típusa:")]
    [Required(ErrorMessage = "A jármű váltójának kiválasztása kötelező!")]
    public int valtoId { get; set; }
    public Valto Valto { get; set; }

    [Display(Name = "A jármű üzemanyagának típusa:")]
    [Required(ErrorMessage = "A jármű üzemanyagának kiválasztása kötelező!")]
    public int UzemanyagId { get; set; }
    public Uzemanyag Uzemanyag { get; set; }

    [Display(Name = "Az ülések/férőhelyek száma:")]
    [Required(ErrorMessage = "Az ülések/férőhelyek számának megadása kötelező!")]
    public int Ules { get; set; }

    [Display(Name = "Az ajtók száma:")]
    [Required(ErrorMessage = "Az ajtók számának megadása kötelező!")]
    public int Ajtok { get; set; }

    [Display(Name = "A gyártás éve:")]
    [Required(ErrorMessage = "A gyártás évének megadása kötelező!")]
    public int GyartasEve { get; set; }

    [Display(Name = "Kép: ")]
    public string Kep { get; set; }

    public byte Elerheto { get; set; }

    [Display(Name = "Bérlés ára/nap: ")]
    [Required(ErrorMessage = "A napi ár megadása kötelező!")]
    public int Ar { get; set; }
}
```

## 5. ábra: Jármű modell

A 6. ábrán lévő Jármű modell váltóra és az üzemanyagra vonatkozó kódrészletei garantálják azt, hogy a Járművek adatbázistábla létrejöttkor, a ValtoId és az UzemanyagId tulajdonságok, mint idegen kulcsok hivatkozzanak a Váltó és Üzemanyag modellekben lévő elsődleges kulcsokra.

A Váltó és Üzemanyag modellek id és név tulajdonságokat tartalmaznak.

```

[Table("Szerzodesek")]
public class Szerzodes
{
    public int Id { get; set; }

    [Required]
    public int UgyfelId { get; set; }
    public Ugyfel Ugyfel { get; set; }

    [Required]
    public int JarmuId { get; set; }
    public Jarmu Jarmu { get; set; }

    [Display(Name = "Bérlés kezdetének időpontja:")]
    [Required(ErrorMessage = "Kérlek add meg a bérlés kezdő időpontját!")]
    [MinKezdo]
    public DateTime BerlesKezdoIdopont { get; set; }

    [Display(Name = "Bérlés végének időpontja:")]
    [Required(ErrorMessage = "Kérlek add meg a bérlés záró időpontját!")]
    [MinZaro]
    public DateTime BerlesZaroIdopont { get; set; }

    public int ArPerNap { get; set; }
    public int Koltseg => (BerlesZaroIdopont.Date.Day - BerlesKezdoIdopont.Date.Day) * ArPerNap;

    public byte Jovahagy { get; set; }
}

```

## 6. ábra Szerződés modell

A Szerződések modell „Jovahagy” tulajdonsága a későbbiekben meghatározza azt, hogy egy szerződéses igény jóváhagyásra váró státuszban van, vagy jóváhagyásra került.

A bérlés kezdő időpontja és záró időpontja esetében feltüntetett [MinKezdo] és [MinZaro] egyedi validációk biztosítják azt, hogy a szerződés kezdő és záró időpontja az aktuális napnál ne eshessen korábbi dátumra valamint azt, hogy a bérlési idő minimum 1 nap legyen.

```

{
    [Table("ArchivaltSzerzodesek")]
    public class Archivum
    {
        public int Id { get; set; }

        public int SzerzodesId { get; set; }

        public string UgyfelNeve { get; set; }
        public string UgyfelCime { get; set; }
        public DateTime? UgyfelSzuletesiDatum { get; set; }
        public string UgyfelJogositvany { get; set; }
        public string UgyfelTelefonszam { get; set; }
        public string UgyfelEmail { get; set; }

        public string JarmuNev { get; set; }
        public string JarmuRendszam { get; set; }
        public string JarmuAlvazsazam { get; set; }
        public int JarmuGyartasEve { get; set; }

        public DateTime BerlesKezdoIdopont { get; set; }
        public DateTime BerlesZaroIdopont { get; set; }

        public int BefizetettOsszeg { get; set; }
        public string Megjegyzes { get; set; }
    }
}

```

## 7. ábra: Archivum modell

Az Archivum modell és az ArchiváltSzerződések adatbázistábla létrehozására azért volt szükség, hogy a jármű bérlésekor fennálló aktuális állapotot (ügyfél személyes adatai, jármű adatok, bérléssel és számlázással kapcsolatos adatok) bármikor reprezentálni tudjuk (szerződés, számlázás), azok ne az aktuális állapotnak megfelelően kerüljenek kiolvasásra az adatbázisból.

Létrehozásra került egy RoleName modell is, mely a különböző jogosultságok neveit tárolja „constans string” formájában.

### 3.4 Az Entity Framework

Az Entity Framework egy eszköz, amit arra használunk, hogy hozzáférjünk egy adatbázishoz. Más megfogalmazásban egy Object-Relational Mapper (rövidítve ORM).



9. ábra: Az Entity Framework

A DbContext egy átjáró az adatbázisunkhoz. Egy DbContextnek egy vagy több DbSet-je van, ami táblákat reprezentál az adatbázisunkban. LINQ-t használunk, hogy ezek a DbSet-ek létrejöjjenek majd az Entity Framework ezt lefordítja SQL utasításokra. Ez egy kapcsolatot nyit az adatbázisba, átolvassa az adatot, objectet alakít ki belőle (mappolás) és visszaadja a DbSet-nek.

Amikor hozzáadunk, törölünk vagy módosítunk adatokat ebből a DbSet-ből az Entity Framework ezt követi. Amint véglegesítjük a változtatásokat feltölti azokat az adatbázisba.

A Car-RentAll applikációban az alábbi DbSet-eket használjuk:

```
public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
{
    public DbSet<Ugyfel> Ugyfelek { get; set; }
    public DbSet<Valto> Valtok { get; set; }
    public DbSet<Uzemanyag> Uzemanyagok { get; set; }
    public DbSet<Jarmu> Jarmuvek { get; set; }
    public DbSet<Szerzodes> Szerzodesek { get; set; }
    public DbSet<Archivum> ArchivaltSzerzodesek { get; set; }

    public ApplicationDbContext()
        : base("DefaultConnection", throwIfV1Schema: false)
    {
    }

    public static ApplicationDbContext Create()
    {
        return new ApplicationDbContext();
    }
}
```

10. ábra: DbSet-ek



Két megközelítése van az Entity Framework használatának:

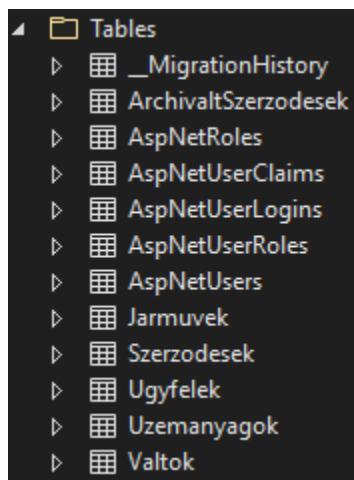
#### 1. Database First

Ebben a megközelítésben előbb az adatbázistáblákat tervezzük meg, majd az Entity Framework legenerálja a szükséges osztályokat.

#### 2. Code First

Ebben a szemléletben az osztályokkal kezdünk és az Entity Framework legenerálja azok alapján az adatbázis tábláit.

Az alkalmazásban az alábbi táblák találhatóak meg:



11. ábra: Adatbázistáblák

A táblák Code First migrációval kerültek létrehozásra.

### 3.2.1 Code First migrációk

A migrációkhoz a NuGet Package Manager Package Manager Consolt használjuk, melyet a project „Tools” menüpontja alatt találunk meg.

A migrációk használata előtt engedélyeznünk kell őket:

- Parancs: enable-migrations

A következő lépés az azonosításhoz használatos táblák, a kezdő állapot létrehozása:

- Parancs: add-migration InitialModel
- Parancs: update-database

A továbbiakban a modelben létrejövő módosításokat az add-migration „MigrációElnevezés” és az update-database parancs páros segítségével fel tudjuk vinni az adatbázisba.

## 3.5 Vezérlők

A webalkalmazásban az alábbi vezérlők találhatók meg:

- Ügyfelek vezérlő
- Járművek vezérlő
- Szerződések vezérlő

A dokumentáció terjedelmi korlátjai és az Ügyfelek, valamint a Járművek vezérlők hasonlósága miatt a két vezérlőt egyben jellemezzük az alábbiakban.

### ApplicationDbContext

A vezérlők közös tulajdonsága, hogy az adatbázisban tárolt adatokhoz a DbContext-en keresztül férnek hozzá. A vezérlők ezért az alábbi sorokkal indulnak:

```
private ApplicationDbContext _context;

public UgyfelekController()
{
    _context = new ApplicationDbContext();
}

protected override void Dispose(bool disposing)
{
    base.Dispose(disposing);
}
```

12. ábra: DbContext

## Index Action

```
public ActionResult Index()
{
    if (User.IsInRole("CanManage"))
        return View("AdminLista");
    else
    {
        var meghivoUgyfel = _context.Ugyfelek
            .SingleOrDefault(u => u.Email == User.Identity.Name);
        if (meghivoUgyfel is null)
            return RedirectToAction("UjUgyfel", "Ugyfelek");
        else
            return View("UgyfelLista", meghivoUgyfel);
    }
}

public ActionResult Index()
{
    var jarmuvek = _context.Jarmuvek
        .Include(v => v.Valto)
        .Include(u => u.Uzemanyag)
        .ToList();
    if (User.IsInRole("CanManage"))
        return View("AdminLista");
    else
        return View("BerelhetoJarmuvek", jarmuvek);
}
```

13. ábra: Index Action

A 13. ábrán látható az Eager Loading. Ez az a jelenség, amikor egy adatbázistáblát a vele kapcsolatban lévő egyéb táblával vagy táblákkal kapcsolatosan hívunk meg. Az ilyen összekapcsolás esetén a nézetben hivatkozhatunk a kapcsolt táblában lévő adatokra. Az Eager Loading használatához szükséges a System.Data.Entity névtér használata.

Mindkét indexben látható, hogy az Index Action meghívásakor eltérő nézetek jelennek meg annak függvényében, hogy a meghívó felhasználó milyen jogosultságokkal bír.

- Amennyiben a felhasználó admin jogosultsággal rendelkezik, az Ügyfelek vezérlő esetében megjelenik minden ügyfél minden adata, melyeket szerkeszteni, törölni lehet, de lehetőség nyílik új ügyfél hozzáadására is. A Járművek esetében az adatbázisban lévő járművek minden adatát látja, valamint itt is lehetőség van az adatok módosítására, járművek törlésére és új járművek hozzáadására.
- Ha a felhasználó nem rendelkezik admin jogosultsággal a saját adatait látja, melyeket módosítani tud, valamint a bérelhető járműveket bootstrap kártyák formájában.

Az Ügyfelek vezérlő azt, hogy egy ügyfél már rendelkezik-e a nyilvántartásban rögzített saját adatokkal, a regisztrációs e-mail cím alapján dönti el. Amennyiben nem adta még meg a saját adatokat, egy üres form-ra kerül átirányításra az Index Action meghívásakor, ahol megadhatja azokat. Amennyiben már megadásra kerültek a saját adatok, lehetősége van azok szerkesztésére.

A szerződések esetében hasonló az Index Action. Amennyiben a felhasználó admin jogosultságokkal rendelkezik, látja a folyamatban lévő, a jóváhagyásra váró és az archivált szerződéseket is. A jóváhagyásra várókat javítani tudja, törölni vagy jóváhagyni, a folyamatban

lévőknek pedig az adatait megtekintheti. Az archivált szerződések esetében a bérlelőkor megadott aktuális állapotokat láthatjuk. A későbbiekben tervezzük elkészíteni a szerződések archivált változatát és a számlát, melyet ebből a nézetből meg tudunk majd jeleníteni.

Amennyiben a felhasználó nem rendelkezik admin jogosultsággal, akkor a saját szerződéseit látja. Jóváhagyásra váró állapotban igényét vissza tudja vonni, saját adatait pedig szerkeszteni tudja. A folyamatban lévő bérlések adatainak és a korábbi bérlések adatainak megjelenítésére is van lehetőség.

## Új jármű és ügyfél hozzáadása

```
[Authorize(Roles = RoleName.CanManage)]
public ActionResult UjJarmu()
{
    var jarmu = new Jarmu { Id = 0 };
    var viewModel = new JarmuValtoUzemanyag
    {
        Jarmu = jarmu,
        Valto = _context.Valtok.ToList(),
        Uzemanyag = _context.Uzemanyagok.ToList()
    };
    return View("JarmuForm", viewModel);
}

[Authorize]
public ActionResult UjUgyfel()
{
    var ugyfel = new Ugyfel { Id = 0 };
    return View("UgyfelForm", ugyfel);
}
```

14. ábra: Új jármű és ügyfél hozzáadása

Az új járművek hozzáadásához admin jogosultság szükséges, de új ügyfél hozzáadásához elegendő, ha a felhasználó be van jelentkezve. Ezeket a tulajdonságokat az alkalmazott attribútumok garantálják.

Mindkét esetben új objektumot hozunk létre, majd az id-t nullára állítjuk. Erre a későbbi Mentés Action megfelelő működése miatt van szükség. Az ügyfél példányt ezután átadjuk a Nézetnek, de a járművek esetében egy ViewModel kerül átadásra, mely tartalmazza a jármű példányt és a váltók, valamint az üzemanyagok listáját. Erre a legördülő listák megfelelő működése miatt van szükség.

A Szerződések vezérlőben az új szerződés létrejötte előtt az ügyfélnek lehetősége van leellenőrizni saját és a kiválasztott jármű adatait. A saját adatait módosítani is tudja. A szerződés kötelező eleme a bérlet kezdő és záró dátumának megadása. A mentés gombra kattintva az igény benyújtásra kerül jóváhagyásra.

## Szerkesztés Action

```
[Authorize(Roles = RoleName.CanManage)]
public ActionResult Szerkesztes(int id)
{
    var jarmu = _context.Jarmuvek
        .Include(v => v.Valto)
        .Include(u => u.Uzemanyag)
        .SingleOrDefault(j => j.Id == id);

    if (jarmu is null)
        return HttpNotFound();

    var viewModel = new JarmuValtoUzemanyag()
    {
        Jarmu = jarmu,
        Valto = _context.Valtok.ToList(),
        Uzemanyag = _context.Uzemanyagok.ToList()
    };

    return View("JarmuForm", viewModel);
}

[Authorize]
public ActionResult Szerkesztes(int id)
{
    var ugyfel = _context.Ugyfelek.SingleOrDefault(u => u.Id == id);
    if (ugyfel is null)
        return HttpNotFound();
    else
    {
        return View("UgyfelForm", ugyfel);
    }
}
```

15. ábra: Járművek és ügyfelek szerkesztése

A járműveket csak admin jogosultsággal rendelkező felhasználó szerkesztheti, míg az ügyféladatok szerkesztése bejelentkezéshez kötött.

Szerződéses adatok szerkesztésére jóváhagyás után nincs lehetőség.

## Mentés Action

```
[Authorize]
[ValidateAntiForgeryToken]
[HttpPost]
public ActionResult Mentés(Ugyfel ugyfel)
{
    if (!ModelState.IsValid)
    {
        if (User.IsInRole("CanManage"))
            return View("UgyfelForm", ugyfel);
        else
            return View("UgyfelLista", ugyfel);
    }

    if (ugyfel.Id == 0)
    {
        _context.Ugyfelek.Add(ugyfel);
    }
    else
    {
        var ugyfelInDb = _context.Ugyfelek.Single(u => u.Id == ugyfel.Id);

        ugyfelInDb.Nev = ugyfel.Nev;
        ugyfelInDb.Cim = ugyfel.Cim;
        ugyfelInDb.SzuletesiDatum = ugyfel.SzuletesiDatum;
        ugyfelInDb.Jogositvany = ugyfel.Jogositvany;
        ugyfelInDb.Telefonszam = ugyfel.Telefonszam;
        ugyfelInDb.Email = ugyfel.Email;
    }

    _context.SaveChanges();

    if (User.IsInRole("CanManage"))
        return RedirectToAction("Index", "Ugyfelek");
    else
        return View("UgyfelLista", ugyfel);
}

[Authorize(Roles = RoleName.CanManage)]
[ValidateAntiForgeryToken]
[HttpPost]
public ActionResult Mentés(Jarmu jarmu)
{
    jarmu.Elerheto = jarmu.Keszlet;
    if (!ModelState.IsValid)
    {
        var viewModel = new JarmuValtoUzemanyag()
        {
            Jarmu = jarmu,
            Valto = _context.Valtok.ToList(),
            Uzemanyag = _context.Uzemanyagok.ToList()
        };

        return View("JarmuForm", viewModel);
    }

    if (jarmu.Id == 0)
    {
        _context.Jarmuvek.Add(jarmu);
    }
    else
    {
        var jarmuInDb = _context.Jarmuvek.Single(j => j.Id == jarmu.Id);

        jarmuInDb.Nev = jarmu.Nev;
        jarmuInDb.Rendszam = jarmu.Rendszam;
        jarmuInDb.Alvazszam = jarmu.Alvazszam;
        jarmuInDb.valtoId = jarmu.valtoId;
        jarmuInDb.UzemanyagId = jarmu.UzemanyagId;
        jarmuInDb.Ajtok = jarmu.Ajtok;
        jarmuInDb.Ar = jarmu.Ar;
        jarmuInDb.GyartasEve = jarmu.GyartasEve;
        jarmuInDb.Keszlet = jarmu.Keszlet;
        jarmuInDb.Elerheto = jarmu.Elerheto;
        jarmuInDb.Kep = jarmu.Kep;
    }

    _context.SaveChanges();

    return RedirectToAction("Index", "Jarmuvek");
}
```

16. ábra: Mentés Action

Ügyfelek esetében a Mentés Action-t bejelentkezett felhasználó használhatja, míg a járművek esetében admin jogosultság szükséges hozzá.

A cross-site request forgery, más néven one-click attack, session riding, rövidítve CSRF vagy XSRF (magyar fordításban kb. oldalon-keresztüli kérés-hamisítás), egy exploittípus, ahol a

weboldalnak küldenek nem-autorizált parancsot a felhasználóként, amelyben megbízik az oldal.[4]

A token használatával garantálhatjuk azt, hogy a támadó távoli elérést használva ne tudjon a felhasználó nevében kéréseket intézni a szerver felé. Az csak a felhasználó számítógépéről lehetséges, mivel ott tárolódik a token titkosítva. Távolról a rejtett mezők nem elérhetők.

Mindkét esetben az Anti-Forgery Token validálásával és a metódus ellenőrzésével kezdünk. Fontos, hogy a mentés csak akkor fut le, ha a token validálása sikeres és a meghívó metódus „POST”.

Az Ügyfelek és a Járművek vezérlők esetében is a modell példányának validálásával kezdünk. Egy modell példánya akkor valid, ha minden kötelező mezője kitöltésre kerül. A validációs üzeneteket az alkalmazásban felhasználói oldalon kezeljük. A kötelező mezők kihagyásával a modell [Required(ErrorMessage = "Üzenet szövege")] attribútumában szereplő üzenet kerül kijelzésre a felhasználó részére.

A 14. ábrán, az új ügyfelek és járművek inicializálásnál láthatjuk, hogy az új példány id azonosítóját 0-ra állítjuk. A leírásban jelezzük, hogy ez a beállítás azért szükséges, hogy a Mentés Action helyesen működjön. Amennyiben a nézet által, a Mentés Action számára átadott példány azonosítója 0, a példány hozzáadásra kerül az adatbázishoz új ügyfél vagy új jármű formájában. Amennyiben az átadott példány már rendelkezik id azonosítóval, a form kitöltésekor megadott adatok felülírják az adatbázistáblában nyilvántartott azonos id-val rendelkező ügyfél vagy jármű tárolt adatait.

A változások elmentése után az Ügyfelek vezérlő esetében a felhasználó annak függvényében, hogy rendelkezik-e admin jogosultsággal visszairányításra kerül a saját adataihoz, vagy az ügyfelek adataihoz. A járművek esetében, mivel a mentés csak admin jogosultsággal használható a járművek adataihoz kerülünk átirányításra.

## **Részletek Action**

```
public ActionResult Reszletek(int id)
{
    var jarmu = _context.Jarmuvek
        .Include(v => v.Valto)
        .Include(u => u.Uzemanyag)
        .Single(j => j.Id == id);

    return View("Reszletek", jarmu);
}
```

17. ábra: Részletek Action

A Járművek vezérlő rendelkezik még egy Részletek Actionnel, mely segítségével a jármű adatok jeleníthetők meg egy nem szerkeszthető mezőkkel ellátott Nézetben. Az ide vezető gombok a bootstrap kártyákon kerültek elhelyezésre.

## 3.6 Nézetek

Egy-egy vezérlőhöz több nézet tartozik. Az, hogy milyen nézet jelenjen meg, a meghívott Action-ön kívül az is befolyásolja, hogy a felhasználó milyen jogosultságokkal rendelkezik.

Az Index Action meghívásakor megjelenő nézeteket az Index Action leírásánál részletezzük.

A nézetben, ha csak „Model”-ként akarnánk hivatkozni az átadott modellre, nem érnénk azt el, mivel a „Model” egy dinamikus struktúra. Ezért a nézet elején pontosítanunk kell, hogy milyen modellt szeretnénk itt használni. A pontosítás után a nézet @Model tagja ismeri, hogy pontosan melyik modelltől van szó és a továbbiakban elérjük azok tulajdonságait.

A formok működését az új járművek hozzáadásához írt form néhány részén keresztül mutatjuk be.

```
using (Html.BeginForm("Action", "Vezérlő"))
{
    <div class="form-group">
        @Html.LabelFor(j => j.Jarmu.Nev)
        @Html.TextBoxFor(j => j.Jarmu.Nev, new { @class = "form-control" })
        @Html.ValidationMessageFor(j => j.Jarmu.Nev)
    </div>

    <div class="form-group">
        @Html.LabelFor(j => j.Jarmu.valtoId)
        @Html.DropDownListFor(j => j.Jarmu.valtoId, new SelectList(Model.Valto, "Id", "Nev"), "Válassz ki a váltó típusát", new { @class = "form-control" })
        @Html.ValidationMessageFor(j => j.Jarmu.valtoId)
    </div>

    @Html.HiddenFor(j => j.Jarmu.Id)
    @Html.AntiForgeryToken()

    <button type="submit" class="btn btn-primary">Mentés</button>
}
}
```

18. ábra: Formok

A form kezdetén meg kell adnunk, hogy a submit típusú gomb lenyomásakor melyik vezérlő, melyik Action-jéhez kerüljön elküldésre a form kitöltésekor megadott adatcsomag.

Az egy-egy tulajdonsághoz tartozó adatokat form-csoportokban kezeljük. Egy ilyen csoport tartalmazza a mező nevét (LabelFor()), egy bemeneti mezőt, ahol meg tudjuk adni a modell adott tulajdonságának értékét az aktuális példány esetében (TextBoxFor()) és egy validációs mezőt, ahol a hibaüzenetek jelennek meg hibás kitöltés esetén.

A mezők neve esetében itt az az érték jelenik meg, amit a modell létrehozásakor a [Display(Name=„MezőNeve”)] annotációban megadtunk. A hibaüzenet szövege a [Required(ErrorMessage=„HibaüzenetSzövege”)] annotációban megadott üzenet.

A @Html egy úgynevezett „helper” metódus. Azért előnyös a használata, mert így a címkére való kattintással is aktívvá válik a hozzá tartozó bemeneti mező, valamint a használatával biztosítjuk azt, hogy a validációs kritériumoknak megfeleljen az új példány is.

A legördülő listák bemeneti mezőjének több paramétere is van:

- 1) A modellben lévő idegen kulcs.
- 2) A lista, amire az idegen kulcs hivatkozik. Mint ahogy fentebb részleteztük, az idegen kulcs egy másik, kapcsolt táblában elsődleges kulcs. Itt adjuk meg azt, hogy a kapcsolt táblában lévő elsődleges kulcshoz milyen megnevezés tartozzon. A legördülő listában a

megnevezések fognak megjelenni, de a form elküldésekor a megnevezéshez tartozó érték kerül majd elküldésre a megfelelő Actionhoz.

- 3) A „placeholder”. Ez az üzenet jelenik meg a legördülő listában, ha még nem került kiválasztásra érték.
- 4) Anonim object.

A form végén rejtett mezőként tároljuk a jármű id-t is, mely új jármű esetében a fent részletezett okok miatt 0, meglévő jármű esetében pedig egy meghatározott érték.

Itt kerül beiktatásra az Anti-forgery token is, melyet a megfelelő Action ellenőriz.

Az ügyfél oldali validációhoz az alábbi script használata szükséges a form végén:

```
@section scripts
{
    @Scripts.Render("~/bundles/jqueryval")
}
```

19. ábra: Ügyfél oldali validáció

Egyedi validációk esetében ügyfél oldali validáció nem lehetséges. Azok mindig szerver oldalon kerülnek ellenőrzésre.

## 3.7 API és a RESTful konvenció

Amikor egy kérés érkezik az applikációtól, az MVC keretrendszer azt egy Actionbe teszi, ami egy vezérlőben van. Ez az Action egy nézettel tér vissza. A RazorView Engine segítségével a szervertől egy HTML markup megy vissza a kliens felé. A HTML markup alapesetben a szerveren készül el, de elkészíthető kliens oldalon is. Ehhez szükséges az API (Application Programming Interface). Az API-k segítségével nyers adatot tudunk a szerveroldalról küldeni a kliens felé, ahol a HTML markup legenerálódik.

### Az API használat előnyei:

- Kevesebb a felhasznált szerver erőforrás (ezáltal magasabb a skálázhatóság)
- Kevesebb a kötődés (ezáltal magasabb a teljesítmény)
- Ügyfelek szélesebb rétegét támogatja (web kliens, mobilapp, tablet)

A REST (Representational State Transfer) egy szoftverarchitektúra típus, elosztott kapcsolat (loose coupling), nagy, internet alapú rendszerek számára, amilyen például a világháló. Azokat a rendszereket, amelyek eleget tesznek a REST megköveteléseinek, "RESTful"-nak nevezik. [5]

REST megkövetelések:

- Kliens-szerver architektúra
- Állapotmentesség
- Gyorsítótárazhatóság



- Réteges felépítés
- Igényelt kód (opcionális)
- Egységes interfész

### 3.8 DTO és az AutoMapper

A RESTful konvenció értelmében a megtervezett modellek közvetlenül nem kapcsolódhatnak az API-hoz. Emiatt úgynevezett Data-Transfer-Objecteket (DTO) használunk az API-k megtervezésénél. A DTO adattovábbításra használatos a kliens és a szerver közt.

Az AutoMapper biztosítja a domain model és a DTO közti mappolást.

Alkalmazásunkban 5 DTO-t különböztetünk meg:

- Ügyfél DTO
- Jármű DTO
- Szerződés DTO
- Váltó DTO
- Üzemanyag DTO

A mappolás az alábbi profil szerint történik:

```
public MappingProfile()
{
    Mapper.CreateMap<Ugyfel, UgyfelDTO>();
    Mapper.CreateMap<UgyfelDTO, Ugyfel>();
    Mapper.CreateMap<Jarmu, JarmuDTO>();
    Mapper.CreateMap<JarmuDTO, Jarmu>();
    Mapper.CreateMap<Valto, ValtoDTO>();
    Mapper.CreateMap<ValtoDTO, Valto>();
    Mapper.CreateMap<UzemanyagDTO, Uzemanyag>();
    Mapper.CreateMap<Uzemanyag, UzemanyagDTO>();
}
```

20. ábra: AutoMapper

Annak érdekében, hogy az alkalmazás indulásakor a MappingProfile is be tudjon tölteni, a Global.asax.cs-ben szükséges azt inicializálni.

### 3.9 API-k az alkalmazásban

Az alkalmazásban API-t használunk az ügyfelek, a járművek és a szerződések megjelenítésénél is. Az ügyfelek esetében és a járművek esetében alkalmazott API ugyanazon 5 Action-t valósítja meg, melyek az alábbiak:

- GET – Az összes ügyfél vagy jármű kiválasztása

```
public IActionResult GetJarmuvek()
{
    var jarmuDTOK = _context.Jarmuvek
        .Include(v => v.Valto)
        .Include(u => u.Uzemanyag)
        .ToList()
        .Select(Mapper.Map<Jarmu, JarmuDTO>);

    return Ok(jarmuDTOK);
}

public IActionResult GetUgyfelek()
{
    return Ok(_context.Ugyfelek.ToList()
        .Select(Mapper.Map<Ugyfel, UgyfelDTO>));
}
```

21. ábra: API GET

- GET(int id) – Egy bizonyos ügyfél vagy jármű kiválasztása id alapján

```
public IActionResult GetUgyfel(int id)
{
    var ugyfel = _context.Ugyfelek.SingleOrDefault(u => u.Id == id);

    if (ugyfel == null)
        return NotFound();
    return Ok(Mapper.Map<Ugyfel, UgyfelDTO>(ugyfel));
}

public IActionResult GetJarmu(int id)
{
    var jarmu = _context.Jarmuvek.SingleOrDefault(j => j.Id == id);

    if (jarmu == null)
        return NotFound();
    return Ok(Mapper.Map<Jarmu, JarmuDTO>(jarmu));
}
```

22. ábra: API GET (int id)

- POST – Egy új ügyfél vagy jármű hozzáadása

```
[HttpPost]
public IActionResult CreateUgyfel(UgyfelDTO ugyfelDTO)
{
    if (!ModelState.IsValid)
        return BadRequest();

    var ugyfel = Mapper.Map<Ugyfel, Ugyfel>(ugyfelDTO);
    _context.Ugyfelek.Add(ugyfel);
    _context.SaveChanges();

    ugyfelDTO.Id = ugyfel.Id;

    return Created(new Uri(Request.RequestUri + $"/{ugyfel.Id}"), ugyfel);
}

[HttpPost]
public IActionResult CreateJarmu(JarmuDTO jarmuDTO)
{
    if (!ModelState.IsValid)
        return BadRequest();

    var jarmu = Mapper.Map<Jarmu, Jarmu>(jarmuDTO);
    _context.Jarmuvek.Add(jarmu);
    _context.SaveChanges();

    jarmuDTO.Id = jarmu.Id;

    return Created(new Uri(Request.RequestUri + $"/{jarmu.Id}"), jarmu);
}
```

23. ábra: API POST

- PUT(int id) – Egy meglévő ügyfél vagy jármű szerkesztése

```
public IActionResult UpdateUgyfel(int id, UgyfelDTO ugyfelDTO)
{
    if (!ModelState.IsValid)
        return BadRequest();

    var ugyfelInDb = _context.Ugyfelek.SingleOrDefault(u => u.Id == id);

    if (ugyfelInDb == null)
        return NotFound();

    Mapper.Map(ugyfelDTO, ugyfelInDb);
    _context.SaveChanges();

    return Ok();
}

public IActionResult UpdateJarmu(int id, JarmuDTO jarmuDTO)
{
    if (!ModelState.IsValid)
        return BadRequest();

    var jarmuInDb = _context.Jarmuvek.SingleOrDefault(j => j.Id == id);

    if (jarmuInDb == null)
        return NotFound();

    Mapper.Map(jarmuDTO, jarmuInDb);
    _context.SaveChanges();

    return Ok();
}
```

24. ábra: API PUT (int id)

- DELETE(int id) – Egy adott ügyfél vagy jármű törlése

```
public IActionResult DeleteUgyfel(int id)
{
    var ugyfelInDb = _context.Ugyfelek.SingleOrDefault(u => u.Id == id);

    if (ugyfelInDb == null)
        return NotFound();

    _context.Ugyfelek.Remove(ugyfelInDb);
    _context.SaveChanges();

    return Ok();
}

[HttpDelete]
public IActionResult DeleteJarmu(int id)
{
    var jarmuInDb = _context.Jarmuvek.SingleOrDefault(j => j.Id == id);

    if (jarmuInDb == null)
        return NotFound();

    _context.Jarmuvek.Remove(jarmuInDb);
    _context.SaveChanges();

    return Ok();
}
```

25. ábra: API DELETE (int id)

Mindkét API vezérlő esetében csak admin jogosultsággal rendelkező felhasználó fér hozzá az API Actionökhöz.

Az API-k úgynevezett státuszkódokat és nyers adatot adnak vissza. A mi esetünkben az alábbi kettő státuszkódot:

- 200 – OK
- 201 – Létrehozva (Created)

A visszaadott nyers adatot az alkalmazásban jQuery-DataTables beépülő modullal alakítjuk át a felhasználói oldalra. A nyers adatot a beépülő modul használatával táblázatba rendezzük, melynek oszlopai rendezhetőek, valamint rendelkezik kereső és oldalkezelő felülettel is. A táblázatban lévő gombokat is felhasználói oldalra generáljuk le.

Általánosságban egy API segítségével feltöltött, felhasználói oldalra legenerált, jQuery-DataTable az alábbiak szerint épül fel:

```
<script>
$(document).ready(function () {
    $("#táblaId").DataTable({
        ajax: {
            url: "/api/vezérlő",
            dataSrc: ""
        },
        columns: [
            {
                data: "tulajdonság neve",
                render: function (data, type, táblaEgyesSzám) {
                    return "<a href='/Controller/Action/' + táblaEgyesSzám.id + '>' + táblaEgyesSzám.nev + "</a>";
                }
            },
            {
                data: "tulajdonság neve"
            },
            {
                data: "id",
                render: function (data) {
                    return "<button class='btn-link js-delete' data-táblaEgyesSzám-id=" + data + ">Törlés</button>";
                }
            }
        ]
    });
});
</script>
```

26. ábra: jQuery-DataTable

A törlés gombot működtető script az alábbiak szerint épül fel általánosságban:

```
$("#TáblaId").on("click", ".js-delete", function () {
    var btn = $(this);
    if (confirm("Megerősítendő üzenet")) {
        $.ajax({
            url: "/api/vezérlő/" + btn.attr("data-táblaEgyesSzám-id"),
            method: "DELETE",
            success: function () {
                table.row(btn.parents("tr")).remove().draw();
            }
        });
    }
});
```

27. ábra: Törlés script

A Szerződés vezérlő API csak egy Action-t tartalmaz, mely segítségével egy szerződést tudunk törölni. Ez a funkció jóváhagyásra váró szerződések esetében használható.

Az API Action az alábbiak szerint épül fel:

```

[HttpDelete]
public IActionResult DeleteSzerzodes(int id)
{
    var szerzodesInDb = _context.Szerzodesek.SingleOrDefault(sz => sz.Id == id);

    if (szerzodesInDb == null)
        return NotFound();

    _context.Szerzodesek.Remove(szerzodesInDb);
    _context.SaveChanges();

    return Ok();
}

```

28. ábra: Szerződés vezérlő, API DELETE

## 3.10 Bejelentkezés és hitelesítés

Mint azt fent részleteztük, a project létrehozásánál az „Individual user accounts” autentikációs lehetőséget választottuk ki. A migrációk engedélyezése után lefuttatott „add-migration InitModel” parancs létrehozza az azonosításhoz használatos táblákat, amit az update-database utasítás után fel is tölt az adatbázisba.

A jogosultságellenőrzést és a jogosultságok szerinti szűrést annotációkkal végezzük el.

Lehetőségünk van globálisan is beállítani autorizációs attribútumot, de ezesetben ott, ahol nem szükséges bejelentkezés, fel kell tüntetnünk az [AllowAnonymous] attribútumot.

A mi webalkalmazásunk esetében globálisan állítottunk be autorizációs attribútumot, majd a Home vezérlőben megadtuk az [AllowAnonymous] attribútumot, így az bejelentkezés nélkül is elérhető.

```

public class FilterConfig
{
    public static void RegisterGlobalFilters(GlobalFilterCollection filters)
    {
        //Saját hozzáadottak
        filters.Add(new AuthorizeAttribute());
        filters.Add(new RequireHttpsAttribute());

        filters.Add(new HandleErrorAttribute());
    }
}

```

29. ábra: FilterConfig

A jogosultságellenőrzés a fenti példákban részletezett attribútumok segítségével történik.

## 3.11 OAuth

A Facebook, Google, Twitter és egyéb külsős oldalak OAuth Autentikációs protokollt használnak (Open Authorisation).

A külső hitelesítés használatához alkalmazásunkat regisztrálnunk kell a Facebookra, hogy egyfajta „partneri kapcsolat” legyen közöttünk. A Facebook a regisztráció után biztosít számunkra egy API-Key-t és egy secretet, ami hasonló, mint egy felhasználónév-jelszó páros. Ezt használjuk, hogy „kommunikáljunk” a Facebookkal. Amikor a felhasználó a facebookos bejelentkezésre kattint, az

alkalmazásunk a Key-Secret páros elküldésével jelzi, hogy a felhasználó az alkalmazásunkat Facebookos regisztrációjával szeretné használni.

A felhasználónk bejelentkezik a Facebook felhasználónevével és jelszavával. A mi alkalmazásunk nem tárolja és nem is akarja tárolni ezeket az adatokat. Amint bejelentkezik, a Facebook megjeleníti neki, hogy az alkalmazásunk hozzá akar férni néhány személyes adatához. A felhasználónak az alkalmazás használatához meg kell adnia ezeket az engedélyeket. Az engedélyek megadása után a Facebook kiküldi az alkalmazásunknak az autorizációs tokenet.

A token megérkezése után a mi alkalmazásunk újra elküldi a tokenet a key-secret párossal a Facebooknak. Ezzel lényegében újra megerősítjük azt, hogy a Facebook küldte-e a tokenet. Ha megerősíti, küld egy Access token-t is. A felhasználó ezzel sikeresen bejelentkezett az alkalmazásunkba a Facebookon keresztül.

A külsős oldalak általi bejelentkezéshez engedélyeznünk kell az SSL-t, valamint garantálnunk kell, hogy az oldal csak HTTPS protokollal legyen elérhető a továbbiakban a biztonságos kommunikáció érdekében. Ezt a 29. ábrán látható „filters.Add(new RequireHttpsAttribute())” globális filterrel állítjuk be.

Az alkalmazás Facebook és Google bejelentkezéssel használható.

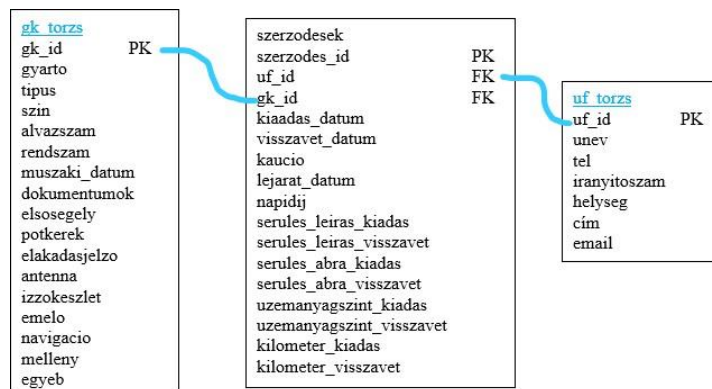
## **4. Az asztali alkalmazás**

A dokumentáció további részében az asztali alkalmazást részletezzük.

### **4.1 Adatbázis**

Kezdetben lokális adatbázist használtunk, majd átköltöztünk az abplusz webtárhelyen foglalt adatbázisára. Az adatbázist ugyan abban a struktúrában építettük fel.

A redundancia elkerülése érdekében 3 táblát hoztunk létre, amelyet másodlagos kulcsokkal kötöttünk össze: gépkocsik, ügyfelek és szerződések táblákat.



30. ábra: Relációs adatmodell

Az **uf\_torzs** tábla kezeli az ügyfelek adatait: azonosítóját, nevét, telefonszámát, lakcímét (irányítószám, település és utca) illetve az emailcímét.

```

create table uf_torzs (

    uf_id                int primary key identity,
    unev                 varchar(40),
    tel                  varchar(20),
    iranyitoszam         int,
    helyseg              varchar(15),
    cim                  varchar(30),
    email                varchar(30),
);
  
```

31. ábra: Ügyféltörzs létrehozókód

A **gk\_torzs** a gépjárművek adatait kezeli: azonosítóját, gyártóját, típusát, színét, alvázszámát, rendszámát, műszaki vizsgájának lejárat dátumát, illetve egyéb feleszereltséggel el van-e látva a gépjármű (pl. láthatósági mellény, pótkerek)

```

create table gk_torzs (

    gk_id                int primary key identity,
    gyarto               varchar(15),
    tipus                varchar(15),
    szin                 varchar(15),
    alvazszam            varchar(30),
    rendszam             varchar(15),
    muszaki_datum        datetime,
    dokumentumok         bit,
    elsosegely           bit,
    potkerek             bit,
    elakadasjelzo        bit,
    antenna              bit,
    radio                bit,
    izzokeszlet          bit,
    emelo                bit,
    navigacio            bit,
    melleny              bit,
    egyeb                varchar(255),
);
  
```

32. ábra: Gépjárműtörzs létrehozókód

A **szerzodesek** tábla a szerződések adatait tartalmazza: azonosítóját, ügyfél azonosítóját, gépjármű azonosítóját, gépjármű kiadásának dátumát, gépjármű várható visszavételének dátumát, az esetleges kauciót, gépjármű kölcsönzés napidíját, a jármű kiadáskor és visszavételekor felírt sérüléseket, üzemanyagszintjét kiadáskor és visszavételkor, illetve a kilométeróra állását kiadáskor és visszavételkor.

```
create table szerzodesek (  
  
    szerzodes_id          int identity,  
    uf_id                 int,  
    gk_id                 int,  
    kiadas_datum          datetime,  
    visszavet_datum       datetime,  
    kaucio                int,  
    lejarat_datum         datetime,  
    napidij               int,  
    serules_leiras_kiadas varchar(255),  
    serules_leiras_visszavet varchar(255),  
    serules_abra_kiadas    tinyint,  
    serules_abra_visszavet tinyint,  
    uzemanyagszint_kiadas  tinyint,  
    uzemanyagszint_visszavet tinyint,  
    kilometer_kiadas      int,  
    kilometer_visszavet   int,  
    primary key (szerzodes_id),  
    foreign key (uf_id) references uf_torzs,  
    foreign key (gk_id) references gk_torzs  
);
```

33. ábra: Szerződések létrehozókód

## 4.2 Bejelentkezés

A bejelentkező felület többször átírássra került. Az elején kettős bejelentkezés volt, mivel a szerverhez is csatlakoznia kellett a felhasználónak. A jelenlegi verzióban csak egyszer kell bejelentkezni az adatbázisba felvett felhasználónévvel és jelszóval.

```

private void btnLogin_Click(object sender, EventArgs e)
{
    try
    {
        var conn = new MySqlConnection(ConnectionString);
        conn.Open();
        var command = new MySqlCommand("select * from felhasznalok;", conn);
        var r = command.ExecuteReader();
        while (r.Read())
        {
            var u = new user();
            u.user_id = r[0].ToString();
            u.username = r[1].ToString();
            u.password = r[2].ToString();
            Users.Add(u);
        }
        r.Close();
        conn.Close();
    }
    catch
    {
        MessageBox.Show("Adatbázis elérése sikertelen, a program kilep!", "Hibaüzenet");
        Properties.Settings.Default.PasswordValid = false;
        Application.Exit();
    }
}

```

34. ábra: Bejelentkezés: try-catch

Try – catch blokk segítségével a program az esetleges hibákat kezeli és kivédi. Hibaüzenetet dob, ha az adatbázisban nem szerepel a beírt jelszó-felhasználónév páros. Az adatbázis kapcsolatot is létrehozza és ellenőrzi a MySqlConnection és MySqlCommand segítségével, és jelzi, ha nem éri el.

```

foreach (user u in Users)
{
    if(u.username == tbUsername.Text)
    {
        if (EncryptDecrypt(tbPassword.Text, 1) == u.password)
        {
            Properties.Settings.Default.UserName=u.username;
            Properties.Settings.Default.PasswordValid = true;
        }
    }
}

```

35. ábra: Bejelentkezés: foreach

A program a foreach ciklussal bejárja, ellenőrzi a bekért adatokat. Ha az adatbázisban szereplő és a bekért adatok egyeznek, belép a program és megjeleníti a Main formot és bezárja önmagát. Hiba esetén üzenetet dob, a try-catch-nak köszönhetően.



```

private List<user> Users = new List<user>();
public string EncryptDecrypt(string szPlainText, int szEncryptionKey)
{
    StringBuilder szInputStringBuild = new StringBuilder(szPlainText);
    StringBuilder szOutStringBuild = new StringBuilder(szPlainText.Length);
    char Textch;
    for (int iCount = 0; iCount < szPlainText.Length; iCount++)
    {
        Textch = szInputStringBuild[iCount];
        Textch = (char)(Textch ^ szEncryptionKey);
        szOutStringBuild.Append(Textch);
    }
    return szOutStringBuild.ToString();
}

```

36. ábra: Encrypt-Decrypt

Az Encrypt-Decrypt egy stringet és egy int kódkulcsot kap, amely stringet ad vissza. Lekódolja a jelszót, az online adatbázis védelme érdekében.

## 4.3 Navigáció

Navigációs ablakot hoztunk létre oldalra, panelek segítségével.

```

private void openChildForm(Form childForm)
{
    if (activeForm != null)
        activeForm.Close();
    activeForm = childForm;
    childForm.TopLevel = false;
    childForm.FormBorderStyle = FormBorderStyle.None;
    childForm.Dock = DockStyle.Fill;
    pnlForm.Controls.Add(childForm);
    pnlForm.Tag = childForm;
    childForm.BringToFront();
    childForm.Show();
}

private void btnSetting_Click(object sender, EventArgs e)
{
    openChildForm(new Setting());
}

```

37. ábra: ChildForm

ChildForm létrehozásával, majd későbbiekben Events-ek által lettek meghívva a formok. Mindig az aktuális childForm látható. Az egységes kinézet, elrendezés érdekében a border stílet levettük, a rendelkezésre álló panelt kitölti.

```

private void btnSettings_Click(object sender, EventArgs e)
{
    frmSettings f = new frmSettings();
    f.ShowDialog();
    SetColor();
}

```

38. ábra: ClickEvent

A childFormoknál előugró ablakokban jelennek meg az új formok a ClickEvent által.

## 4.4 Kereső és szűrő feltételek

```

var conn = new MySqlConnection(connectionString);
conn.Open();
var command = new MySqlCommand(
    "select szerzodes_id, unev, iranyitoszam, helyseg, cim, gyarto, tipus, " +
    "rendszer, kiadas_datum, lejarat_datum, visszavet_datum " +
    "from gk_torzsz, uf_torzsz, szerzodesek " +
    "where " +
    "szerzodesek.uf_id = uf_torzsz.uf_id and " +
    "szerzodesek.gk_id = gk_torzsz.gk_id and " +
    $"{(InsertedString)} " +
    $"unev like '{(name_filter)}';", conn);
var rows = command.ExecuteReader();
while (rows.Read())
{
    string address = rows[2] + " " + rows[3] + " " + rows[4];
    string car = rows[5] + " " + rows[6];

    dgvMain.Rows.Add(
        rows[0], rows[1], address, car,
        rows[7], rows[8], rows[9], rows[10]);
}

conn.Close();
}

```

39. ábra: DGV

Kereső és szűrő feltételek előtt a DataGridView-ban megjelenítjük a kívánt adatokat. A kapcsolat megnyitása után a SQL lekérdezést hajtunk végre. Commandban megadjuk a Select után a látni kívánt táblákat, az összekötő táblákat és a feltételeket. A lekért adatokat soronként a DTG-ba olvassuk.

```

private void tbSearchByType_KeyUp(object sender, KeyEventArgs e)
{
    refreshDgv(tbSearchByType.Text, cbCategories.SelectedIndex);
}

private void cbCategories_DropDownClosed(object sender, EventArgs e)
{
    refreshDgv("", cbCategories.SelectedIndex);
}

```

40. ábra: Kereső és szűrő feltétel

A kereső feltételnél a DGV pontosítja a keresést a TextBoxba gépeltek alapján, míg a szűrő feltétel szűkíti a keresett adatot az adatbázis egyik cellája alapján. Mind a kettő a keresés után frissíti a DataGridView-t.

## 4.5 Új adat hozzáadása

```
private void btnNewDevice_Click(object sender, EventArgs e)
{
    var f = new frmDevicesDataInputTable(-1,ConnectionString);
    f.ShowDialog();
    refreshDgv("", cbCategories.SelectedIndex);
}
```

41. ábra: Input Form meghívása

Új adat bevitelekor a gombra egy Eventet tettünk és egy másik ablakban jelent meg a beviteli mező, form formájában (InputTable).

```
public frmCustomerDataInputForm(string c, int id)
{
    InitializeComponent();
    ConnectionString = c;
    Id = id;

    if (id > 0)
    {
        btnOK.Text = "Mentés";
        using (var conn = new MySqlConnection(ConnectionString))
        {
            conn.Open();
            var command = new MySqlCommand(
                "SELECT * from uf_torzs " +
                $"WHERE uf_id={id}", conn);

            var sor = command.ExecuteReader();

            while (sor.Read())
            {
                tbName.Text = sor[1].ToString();
                tbPhoneNumber.Text = sor[2].ToString();
                tbPostcode.Text = sor[3].ToString();
                tbCity.Text = sor[4].ToString();
                tbaddress.Text = sor[5].ToString();
                tbEmail.Text = sor[6].ToString();
            }
        }
    }
}
```

42. ábra: Adat hozzáadása

A formokon a beviteli mezők szövegdobozok voltak, de alkalmaztunk legördülő listákat és DateTimePickereket is. SQL lekérdezés segítségével lekértük a táblát, ahová az adatbeviteli mezők értékeit beolvastuk.

## 4.6 Adat törlése

```
try
{
    using (var conn = new MySqlConnection(ConnectionString))
    {
        conn.Open();
        var command = new MySqlCommand(
            "DELETE FROM gk_torzs " +
            $"WHERE gk_id = {dgvDeviceList.SelectedRows[0].Cells[5].Value}", conn);

        command.ExecuteNonQuery();
    }
}
catch (MySqlException ex)
{
    if (ex.Number == 1451)
    {
        MessageBox.Show("A gépjármű használatban van (nem törölhető!)");
        return;
    }
}
```

43. ábra: Adat törlése

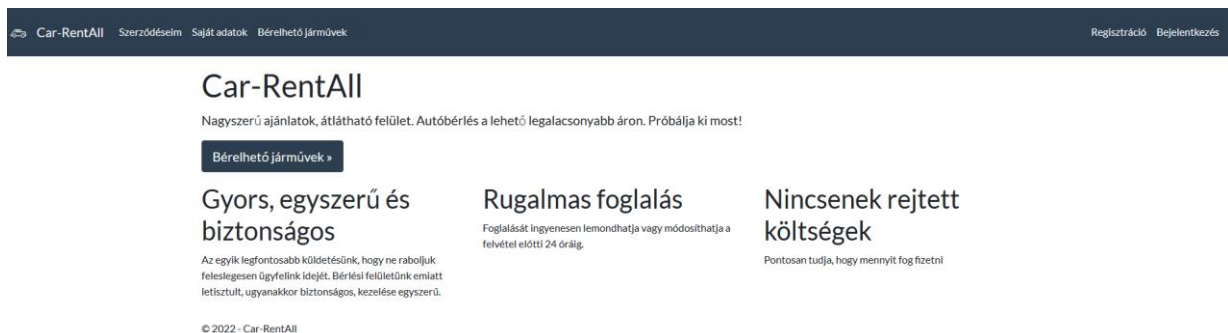
Az adat törlése általában try-catch által történt. A try-nál sikeresen törlődött az adat az adatbázisból, és az executeNonQuery jelzi, hány sort érintett a változás. Catch esetén a hibát jelzi a rendszer.

## 5. Felhasználói dokumentáció

### 5.1 Webalkalmazás

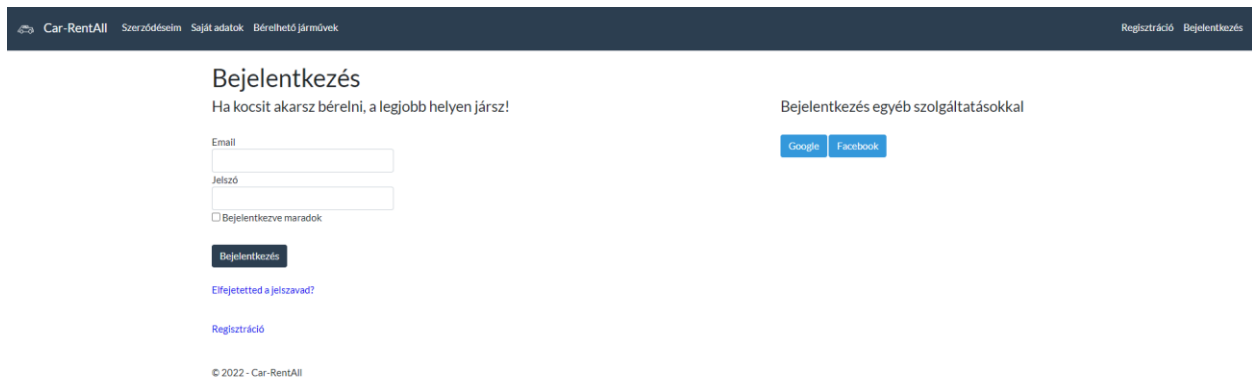
Az alkalmazást bejelentkezés után tudjuk használni.

#### 5.1.1 Főoldal



Főoldal

#### 5.1.2 Bejelentkezés



Bejelentkezés

Ügyfélként emailcím és jelszó megadásával lehet regisztrálni, vagy Facebook, illetve Google szolgáltatással is be lehet jelentkezni.

### 5.1.3 Saját adatok

Az ügyfél regisztráció után megadhatja adatait, illetve későbbiekben módosíthatja azokat.

#### Saját adatok hozzáadása

Név:

Cím:

Születési dátum:

Jogosítvány azonosító:

Telefonszám:

E-mail cím:

Mentés

© 2022 - Car-RentAll

Saját adatok hozzáadása

#### Saját adatok szerkesztése

Név:

Cím:

Születési dátum:

Jogosítvány azonosító:

Telefonszám:

E-mail cím:

Mentés

© 2022 - Car-RentAll

Saját adatok szerkesztése

### 5.1.4 Bérelhető járművek

A bérelhető járművek jelennek meg.

Car-RentAll

Szerződéselm

Saját adatok

Bérelhető járművek


Szla.ugyfel12@gmail.com

Kijelentkezés

Bérelhető Járművek

Renault Clio

Váltó: Manuális  
Üzemanyag: Benzín




Férőhelyek száma: 4  
Ajtók száma: 4  
Gyártás éve: 2000  
Már napi 7500 Ft-tól

Részletes adatok

Árjárlatot kérek

Opel Grandland

Váltó: Manuális  
Üzemanyag: Benzín




Férőhelyek száma: 5  
Ajtók száma: 5  
Gyártás éve: 2006  
Már napi 11200 Ft-tól

Részletes adatok

Árjárlatot kérek

Skoda SuperB

Váltó: Automata  
Üzemanyag: Benzín



Férőhelyek száma: 5  
Ajtók száma: 5  
Gyártás éve: 2011  
Már napi 9500 Ft-tól

Részletes adatok

Árjárlatot kérek

Bérelhető járművek

### 5.1.5 Részletes adatok / Árajánlatot kérek

A jármű részletes adatait nézhetjük át, valamint lehetőségünk van ebben a menüpontban az „Árajánlatot kérek” gombra kattintva egy összesítő ellenőrzésre. Az ügyfél a saját adatait még tudja ezen a képernyőn módosítani.

#### Ford Fiesta részletes adatai



A jármű neve:	Ford Fiesta
A váltó típusa:	Automata
Üzemanyag:	Benzin
Ajtók száma:	5
Férőhelyek száma:	5
Gyártás éve	2001

Árajánlatot kérek

© 2022 - Car-RentAll

#### Részletes adatok

#### Árajánlat áttekintés

##### Saját adatok:

Név:	ÚjÜgyfél10
Cím:	ÚjCím10
Születési dátum:	1994-05-05
Jogosítvány:	ÚjJogsi10
Telefonszám:	06305698754
E-mail cím:	ujugyfel10@gmail.com

Szerkesztés

##### Bérlésre vonatkozó adatok:

Bérlés kezdetének időpontja:
2022. 05. 15.
Bérlés végének időpontja:
2022. 05. 22.

##### A kiválasztott jármű adatai:

<b>Ford Fiesta</b>
Váltó: Automata Üzemanyag: Benzin
Férőhelyek száma: 5
Ajtók száma: 5
Gyártás éve: 2001
Már napi 7000 Ft-tól
<a href="#">Részletes adatok</a> <a href="#">Árajánlatot kérek</a>

Árajánlatot kérek!

#### Árajánlat áttekintés

## 5.1.6 Szerződéseim

A felhasználó láthatja az aktív és már lezárt szerződéseit. Lehetőség van az igény visszavonására amennyiben a szerződés még jóváhagyásra vár. Az archivált szerződések esetében megtekinthető a számla, a szerződés és a bérleskor aktuális ügyfél és jármű adatok, valamint a bérlésre vonatkozó információk is.

### Jóváhagyásra vár:

Jármű	Bérlés kezdő időpontja	Bérlés záró időpontja	Fizetendő díj	Kaució	Státusz	
Ford Fiesta	2022-05-15	2022-05-22	49000 Ft	9800 Ft	Jóváhagyásra vár	
Nissan Navarra	2022-05-15	2022-05-21	108000 Ft	21600 Ft	Jóváhagyásra vár	
Hyundai Tucson	2022-05-15	2022-05-24	180000 Ft	36000 Ft	Jóváhagyásra vár	

### Aktuális, bérelt járművek










Jármű	Bérlés kezdő időpontja	Bérlés záró időpontja	Befizetett bérleti díj	Státusz
Skoda SuperB	2022-05-15	2022-05-20	47500	Jóváhagyva, folyamatban

Korábbi szerződéseim

© 2022 - Car-RentAll

### Szerződés áttekintő

### Korábbi szerződéseim

Azonosító	Szerződéses ügyfél adatok	Bérelt jármű adatok	Bérléshez kapcsolódó adatok	Műveletek
CR/Archiv/24	ÚjÜgyfél10 	Renault Captur 	Bérlés 	 
CR/Archiv/27	ÚjÜgyfél10 	Renault Clio 	Bérlés 	 
CR/Archiv/29	ÚjÜgyfél10 	Ford Fiesta 	Bérlés 	 
CR/Archiv/32	ÚjÜgyfél10 	Renault Captur 	Bérlés 	 
CR/Archiv/40	ÚjÜgyfél10 	Ford Fiesta 	Bérlés 	 

© 2022 - Car-RentAll

### Archivált szerződések

## 5.1.7 Ügyfelek

Admin jogosultsággal rendelkező felhasználók láthatják az alábbi menüpontot.

### Ügyfelek

[+ Új ügyfél hozzáadása](#)

Show 10 entries Search:

Név	Cím	Születési dátum	Jogosítvány	Telefonszám	E-mail		
Am Érika	1121, Budapest, Petoő Sándor utca 12.	1974-05-05	MB1986745	06305987465	ugyfel1@carrentall.com	<a href="#">✎</a>	<a href="#">✖</a>
Para Zita	3000, Hatvan, Fo út 6.	1971-11-29	KN6598732	06209683245	ugyfel3@carrentall.com	<a href="#">✎</a>	<a href="#">✖</a>
Patta Nóra	1125, Budapest, Bartók sétány 4.	1995-05-11	HG6976432	06708567421	ugyfel2@carrentall.com	<a href="#">✎</a>	<a href="#">✖</a>
Remek Elek	8600 Siófok, Balatoni út 46.	1998-09-29	KK6935461	06708962156	ugyfel4@carrentall.com	<a href="#">✎</a>	<a href="#">✖</a>

Showing 1 to 4 of 4 entries Previous 1 Next

© 2022 - Car-RentAll

### Ügyfelek


Az ügyfelek adatait listázza ki, amelyet lehet szerkeszteni és törölni.

Új ügyfél hozzáadása gombbal új ügyfeleket lehet rögzíteni az adatbázisban, az alábbi adatok megadásával: ügyfél neve, címe, születési dátuma, telefonszáma, emailcíme.

### Új ügyfél hozzáadása

Név:

Cím:

Születési dátum:  
 

Jogosítvány azonosító:

Telefonszám:

E-mail cím:

[Mentés](#)

© 2022 - Car-RentAll

### Új ügyfél hozzáadása



## 5.1.8 Járművek

Admin jogosultsággal rendelkező felhasználók számára.

Car-RentAllSzerződésekÜgyfelekJárművek

Szia admin@carrentall.com!Kijelentkezés

### Járművek

Új jármű hozzáadása

Show: 10 entries

Search:

Név	Rendszám	Alvázsám	Váltó	Üzemanyag	Ülések	Ajtók	Gyártás éve	Alap napi ár		
Ford Fiesta	AAA111	FiktívAlvázsám	Automata	Benzín	5	5	2001	0		
Opel Astra	BBB222	FiktívAlvázsám	Manuális	Diesel	4	5	2005	0		
Renault Captur	CCC333	FiktívAlvázsám	Automata	Diesel	5	5	2007	0		
Renault Clio	DDD444	FiktívAlvázsám	Manuális	Benzín	4	4	2000	0		

Showing 1 to 4 of 4 entries

Previous1Next

© 2022 - Car-RentAll

### Járművek

A gépjárműveket listázza ki az alábbi menüpont, ahol a járművek adatait lehet szerkeszteni és törölni.

Új jármű hozzáadása gombbal új gépjárművet lehet rögzíteni az adatbázisban, az alábbi adatok megadásával: jármű neve, rendszáma, alvázsáma, váltó típusa, üzemanyag típusa, férőhelyek száma, ajtók száma, gyártás éve, bérleti díj és képfeltöltésével.

A jármű neve:

A jármű rendszáma:

A jármű alvázsáma:

A jármű váltójának típusa:

A jármű üzemanyagának típusa:

Az ülések/férőhelyek száma:

Az ajtók száma:

A gyártás éve:

Bérlés ára/napi:

Fénykép feltöltése  

Fájl kiválasztása

Nincs fájl...álasztva

Kép feltöltése

Kép:

Mentés

### Új gépjármű

## 5.1.9 Szerződések

Admin jogosultsággal rendelkező felhasználó jóváhagyhatja, szerkesztheti vagy törölheti a szerződéseket.

### Szerződések áttekintő

Jóváhagyásra vár:

Ügyfél	Jármű	Bérlés kezdő időpontja	Bérlés záró időpontja	Fizetendő díj	Kaució	Műveletek
ÚjÜgyfél10	Ford Fiesta	2022-05-15	2022-05-22	49000 Ft	9800 Ft	   
ÚjÜgyfél10	Nissan Navarra	2022-05-15	2022-05-21	108000 Ft	21600 Ft	   
ÚjÜgyfél10	Hyundai Tucson	2022-05-15	2022-05-24	180000 Ft	36000 Ft	   

### Folyamatban lévő bérlések

Ügyfél	Jármű	Bérlés kezdő időpontja	Bérlés záró időpontja	Befizetett bérleti díj	Műveletek
ÚjÜgyfél10	Skoda SuperB	2022-05-15	2022-05-20	47500	 

### Archívum

















Archivált szerződések megtekintése

© 2022 - Car-RentAll

### Szerződések áttekintő

## 5.1.10 Archívum

Az archívumban az archivált szerződéseket és számlákat tekinthetjük meg.

Car-RentAll	Szerződések	Ügyfelek	Járművek	Szia admin@currentall.com! Kijelentkezés
CR/Archiv/24	ÚjÜgyfél10	Renault Captur	Bérlés	 
CR/Archiv/27	ÚjÜgyfél10	Renault Clio	Bérlés	 
CR/Archiv/17	ÚjÜgyfél 4	Renault Captur	Bérlés	 
CR/Archiv/16	ÚjÜgyfél 5	Ford Fiesta	Bérlés	 
CR/Archiv/21	ÚjÜgyfél 6	Ford Fiesta	Bérlés	 
CR/Archiv/29	ÚjÜgyfél10	Ford Fiesta	Bérlés	 
CR/Archiv/32	ÚjÜgyfél10	Renault Captur	Bérlés	 
CR/Archiv/40	ÚjÜgyfél10	Ford Fiesta	Bérlés	 
<div><div><strong>Ügyfél adatai:</strong><ul style="list-style-type: none"><li>Név: ÚjÜgyfél10</li><li>Cím: ÚjCím10</li><li>Születési dátum: 1994-05-05</li><li>Jogosítvány: ÚjJogsi10</li><li>Telefonszám: 06305698754</li><li>E-mail cím: ujgyfe110@gmail.com</li></ul></div><div><strong>Jármű adatai:</strong><ul style="list-style-type: none"><li>Név: Ford Fiesta</li><li>Rendszám: FF Rendszám</li><li>Alvázszám: FFAlvázszám</li><li>Gyártás éve: 2001</li></ul></div></div>				

© 2022 - Car-RentAll

### Archívum

## 5.2 Asztali alkalmazás

A programot bejelentkezés után lehet használni.

### 5.2.1 Bejelentkezés









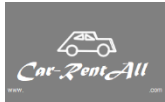
Bejelentkezés

A programot vásárló személynek létrehoztunk egy felhasználót (admin). Bejelentkezést követően további felhasználókat tud létrehozni a Felhasználók menüpont alatt.

### 5.2.2 Navigációs ablak

Bejelentkezést követően megnyílik az alkalmazás, melynek bal oldalán található a navigációs ablak 6 menügombbal:

-  Szerződések – aktuális és lezárt szerződéseket találhatók itt
-  Gépjárművek listája – kint és bent lévő gépjárműveinket is itt találjuk
-  Ügyféltörzs – ügyfeleink adatait láthatjuk, illetve új ügyfeleket adhatunk hozzá a listánkhoz
-  Felhasználók – új felhasználó hozzáadása, esetleg törlése, új jelszó megadása
-  Beállítások – felhasználó testre szabhatja a fiókját
-  Kilépés – bezárja az alkalmazást



Az alkalmazás logójára (szövegre: *Car-RentAll*) kattintva a Car-RentAll honlapja nyílik meg.

### 5.2.3 Szerződések

A szerződések menüpontban az ügyfelek szerződései láthatók. Státusz és ügyfél neve alapján lehet szűrni a listában.

Szerződések Adminisztráció Beállítások Kilépés

Szűrési feltétel kiválasztása  
Minden szerződés

Keresés

Sorszám	Ügyfél neve	Ügyfél címe	Kibérelt eszköz	Rendszám	Kiadás dátuma	Lejárati dátuma	Visszavétel dátuma
15	Kuki Muki	1062 Budapest Messz...	Opel Astra	FDC-387	2022. 05. 01. 1:02:00	2022. 04. 29. 0:00:00	2022. 05. 01. 1:37:00
32	Nagy Peter	1224 Csomor Virag utc...	Renault Capture	FDE-394	2022. 05. 04. 2:07:00	2022. 05. 04. 2:07:00	2022. 03. 14. 2:10:00
33	Kiss Geza	1062 Debrecen Pityan...	Opel Astra	FDC-387	2022. 05. 04. 6:10:00	2022. 05. 20. 6:10:00	
34	Vékony Lilla	1634 Kaposvár Kolbasz...	Renault Capture	FDE-394	2022. 05. 17. 11:52:00	2022. 05. 26. 11:52:00	

#### Szerződések

Szerződés lezárása gombbal a listából kiválasztott szerződést lehet lezárni. Az előugró ablakban meg kell adni a következő adatokat: kilométeróra állása visszavételkor, üzemanyag mennyisége, visszavétel dátuma, esetleges pótdíj mértéke, illetve egyéb megjegyzéseket lehet megadni (pl. sérülések).

Szerződés lezárása

Kilométeróra állása	<input type="text"/>	Sérülések leírása, egyéb megjegyzés
Üzemanyagmennyisége	0%-25% ▾	<div></div>
Visszavétel dátuma	2022-05-14 07:56 📅 ▾	
Esetleges pótdíj összege	0	
<div>Lezár</div> <div>Mégse</div>		

### Szerződés lezárása

Új szerződés menügombbal új szerződést lehet létrehozni, amelyhez az alábbi adatok szükségesek, hogy az adatbázisba rögzíteni tudjuk: ügyfél neve, lakcíme, gépjárműre fizetendő kaució és bérleti díj megadása, kilométerállása és üzemanyag mennyisége kiadáskor, kiadás dátuma és várható lejárata.

Új szerződés készítése

Ügyfél neve Kuki Muki Ügyfél címe 1062 Budapest Messszelinnen utca 54. Szabad gépjármű kiválasztása ▾ Bérleti időszak kezdete 2022-05-14 07:53 📅 ▾ Bérleti időszak várható lejárata 2022-05-14 07:53 📅 ▾ Kilométeróra állása kiadáskor <input type="text"/> Üzemanyag mennyisége kiadáskor 0% - 25% ▾	<div>Ügyfél kiválasztása</div> Bérleti díj <input type="text"/> Kaució <input type="text"/> Sérülések kiadáskor <div></div>
<div>Rögzítés</div>	<div>Mégsem</div>

### Új szerződés

Ügyfél kiválasztása menügombbal az ügyfelek listája jelenik meg.

## 5.2.4 Ügyféltörzs

Opciók Bezár

Új ügyfél felvétele

Szerkesztés

Ügyfél törlése

Bezár / Kiválaszt

Ügyfelek listája

Név szerinti keresés

Név	Irányítószám	Helység	Cím
Kuki Muki	1062	Budapest	Messszelnnen ...
Kiss Geza	1062	Debrecen	Pitypang utca 1.
Malna Peter	1033	Pecs	Kepes utca 13.
Kover Andras	1161	Kiskunhalas	Zoldseg utca 45.
Nagy Peter	1224	Csomor	Virag utca 122.
Vekony Lilla	1634	Kaposvar	Kolbaszos utca...
Almas Pite	1122	Szeged	Szalonnas utc...
Szegedi Szalami	1673	Nyiergyhaza	Akarmilyen utc...
Kolbasz Miklos	1884	Pecs	EzNemAz utca ...
Elem Elemer	1226	Debrecen	MajdnemOttVa...
Repa Zold	1992	Budapest	Vilaqvege utca...

### Ügyfelek listája

Keresési feltétellel szűkíteni tudjuk a találatokat: keresési kategória vagy típus alapján.

A kiválasztott ügyfelet az Ügyfél törlése gombbal lehet törölni az adatbázisból.

Szerkesztés gombbal a meglévő ügyfél adatait lehet módosítani.

Új ügyfélfelvétele gombbal új ügyfelet lehet rögzíteni az adatbázisba, ha az alábbi adatokat megadjuk: ügyfél neve, telefonszáma, irányítószáma, települése, címe és emailcíme.

Ügyféladat

Ügyfél adatai

Név

Telefonszám

Irányítószám

Helység

Cím

Email

Új létrehozása

Mégse

Új ügyfél felvétele

### 5.2.5 Gépjárművek listája

A menüpontban jobb oldalt az adatbázisban rögzített gépjárművek jelennek meg, bal oldalt alul pedig a gépjárművek képei.

www. .com

- Szerződések
- Gépjárművek listája
- Ügyféltörzs
- Felhasználók
- Beállítások
- Kilépés

Szerződések    Adminisztráció    Beállítások    Kilépés

Opciók    Bezárás

Új gépjármű

Gépjármű adataink szerkesztése

Gépjármű törlése

Gépjárművek listája

Kategória: Mindegyik

Keresés típus szerint

Rendszám	Gyártmány	Típus	Alvázsorszám	Kategória
NFR-110	Ford	Fiesta	VV67299LO123	Teherauto
FDC-387	Opel	Astra	VF34LC6799988	Szemelyauto
FDE-394	Renault	Capture	13131231233	Teherauto
DAF-614	Renault	Clio	VVF333HU77853	Szemelyauto

Gépjárművek listája

Az alábbi menüpontban 3 gomb található:

- Gépjármű törlése - a listában kiválasztott gépjárművet törli az adatbázisból
- Gépjármű adatainak szerkesztése - a kiválasztott gépjármű adatait lehet módosítani
- Új gépjármű – új gépjármű adatait lehet rögzíteni az adatbázisban, amelyhez legördülő listában lehet megadni a gépjármű gyártóját, típusát és színét. Ha a listában nincs az általunk keresett adat, a plusz jeles gombra kattintva, majd a listába beírva hozzálehet adni új adatot az adatbázishoz. Adat eltávolítása a mínusz jeles gombbal történik.

Ezen felül a gépjármű rendszámát, alvázszámát, személyek számát, gyártási évet és ajtók számát kell megadni.

Legördülő listából lehet választani a kategória, váltó és üzemanyagtípusánál.

Képet megadni célszerű, de nem kötelező. Opcionális továbbá a megjegyzés és a felszereltség.

Gépjármű adatai

Gyártó	<input type="text"/>	+	-	Kategória	Szemelyauto
Típus	<input type="text"/>	+	-	Váltó	Kezi
Rendszám	<input type="text"/>			Üzemanyag	Benzin
Alvázszám	<input type="text"/>			Személyek száma	<input type="text"/>
Szín	<input type="text"/>	+	-	Gyártási év	<input type="text"/>
Műszaki érvényesség	2022-04-09			Ajtók száma	<input type="text"/>

File: ..\..\Resources\No\_Phography.png

Kép kiválasztása


Felszereltség:

<input type="checkbox"/> Dokumentumok	<input type="checkbox"/> Radió
<input type="checkbox"/> Elsősegély doboz	<input type="checkbox"/> Izzókészlet
<input type="checkbox"/> Pótkerék	<input type="checkbox"/> Emelő
<input type="checkbox"/> Elakadásjelző	<input type="checkbox"/> Navigáció
<input type="checkbox"/> Antenna	<input type="checkbox"/> Láthatósági mellény

Megjegyzés

Mentés

Bezárás



Új gépjármű felvétele



## 5.2.6 Felhasználók

A menüpontban először ki kell választani, hogy milyen műveletet legyen végrehajtva.

- Új ügyfél hozzáadása – gombra kattintva a program engedélyezi, hogy új felhasználónév és jelszó legyen megadva
- Felhasználó törlése – gombra kattintva, ki kell jelölni az oldalsó listából a törölni kívánt felhasználót. Jelszó megadásával törölhető (Kivéve admin)
- Jelszó módosítás – gombbal meg lehet változtatni a jelszót

Felhasználó menüpont

## 5.2.7 Beállítások

A beállítások menüpontban be lehet állítani az alkalmazás módját és színét.

- sötét – háttér fekete, szöveg fehér
- világos – háttér szürkésfehér, szöveg fekete

A módokat a színekkel tudjuk változtatni. A megadott módban a kiválasztott szín szerint az alábbiak módosulnak:

- piros – gombok háttere világosszürke, szöveg piros
- zöld – gombok háttere világoszöld, szöveg zöld
- kék – gombok háttere világos kék, szöveg kék

Az alapértelmezett gombbal a témát alapértelmezettre állítjuk vissza.

Beállítások

Mód kiválasztása

Sötét

Világos

Szín kiválasztása

Piros

Zöld

Kék

Alapértelmezett

Beállítások

## 6. Összegzés

### 6.1 Tesztesetek

A programokat készítés során folyamatosan teszteltük. Csak a projektben résztvevő személyek használták az asztali és a webalkalmazást.

Tesztelések közben hibáink igyekeztünk észrevenni és javítani.

#### 6.1.1 Webalkalmazás

##### ✓ Felhasználókezelés

A felhasználók különböző jogosultságai miatt meg kellett oldanunk, hogy az admin jogosultsággal rendelkező felhasználó teljesen más menürendszert lásson, mint egy normál felhasználó. Ezt két külön navigációs menüvel oldottuk meg.

##### ✓ Bejelentkezés, regisztráció

Admin és felhasználó részéről is hibamentesen zajlott.

##### ✓ Kilépés

Hibamentesen zajlott.

##### ✓ Szerződések

A szerződések esetében meg kellett oldanunk, hogy az aktuális dátumtól korábbi dátumot ne tudjon a felhasználó beállítani kezdő és záró dátumnál, valamint azt, hogy a záró dátum minimum egy nappal a kezdő dátum után legyen. Ezt egyedi validációval oldottuk meg.

El kellett különítenünk a jogosultságok alapján a megjelenő szerződések nézetet.

##### ✓ Ügyfelek listázása

Hibamentesen zajlott.

##### ✓ Járművek listázása

A járművek listázásánál azt kellett megoldanunk, hogy a járművek adataival együtt töltődjenek be a csatolt táblák adatai is.

### **6.1.2 Asztali alkalmazás**

#### **✓ Bejelentkezés**

Hibamentes volt a tesztelés.

#### **✓ Navigáció**

Hibamentesen zajlott.

#### **✓ Szerződések**

Szerződés lezárásánál a kiadás dátumától régebbi dátumot is meglehetett adni. A hibák észrevétele után javítva lettek.

#### **✓ Ügyféltörzs**

A tesztelés hibamentesen zajlott.

#### **✓ Gépjárművek listája**

Kezdetben nem jelenítette meg a képet. Hiba észrevétele után javítva lett.

#### **✓ Beállítások**

Sokáig nem csak a Setting form változott a beállítások következtében, a többi nem. A hibát javítottuk.

#### **✓ Felhasználók**

Bárki, bármilyen felhasználót törölhetett. A hibát észrevettük, és kezeltük. Csak akkor törölhető egy felhasználó, ha megadja jelszavát, illetve az admin nem törölhető.

#### **✓ Kilépés**

Hibamentesen zajlott.

## 6.2 Továbbfejlesztési lehetőségek

Ahogy minden program, úgy a miénk is több továbbfejlesztési lehetőséget rejt magában:

- ❖ Elképzeltük, azonban nem sikerült megvalósítanunk, hogy egy adatbázisról fusson az asztalai és a webes alkalmazás.
- ❖ Továbbfejlesztési lehetőség, hogy panelek segítségével legördülő almenüket hozzunk létre, s így nem lennének „felugró ablakok”.
- ❖ Felhasználók számának bővítése, nagyobb cég esetén.
- ❖ Az asztali alkalmazás bejelentkezés után jelenítse meg a felhasználó nevét.
- ❖ Dolgozók számára e-mailcím megadása.
- ❖ Kiválasztott szerződés nyomtatása.
- ❖ Form alkalmazás során a kiválasztott kinézetet tartsa meg a következő bejelentkezés során is.
- ❖ Beállításokban kilehessen kapcsolni (elrejtteni) a ToolMenüt.
- ❖ A webalkalmazás designja rengeteg továbbfejlesztési lehetőséget rejt magában.

## 6.3 Összegzés

A Car-RentAll egy általunk készített webes és asztali alkalmazás, amely a gépjárműkölcsönzést segíti egy adott cég és ügyfele számára.

Szinte az összes eltervezet funkciót létrehoztuk az alkalmazásunkban. Legnagyobb sajnálatunkra, ami szorosan összekötné a webes és az asztali alkalmazást, az adatbázist nem sikerült összehoznunk. Viszont így, egymástól függetlenül is használható mind a két felület.

Legnehezebb számunkra az ASP létrehozása volt. A projekt elején nem is gondolkodtunk webes alkalmazásban, csak asztaliban. A webeset alkalmazást tanulás az iskolában párhuzamosan tanultak alapján állítottuk össze. Az elején nehéz volt átlátni az új és szokatlan fejlesztői környezetet, azonban az idő előrehaladtával egyre jobban kiismertük magunkat az új fejlesztői környezetben.

A másik nagy kihívás számunkra a csapatban munka volt. Sokszor nehéz volt megfelelő időpontot találni, hogy összeüljünk egyeztetni munkahelyi és/vagy magánéleti elfoglaltságaink miatt. Egy idő után ezzel se volt probléma, és a közös munka gördülékenyen ment. Remek tapasztalat volt számunkra, hogy milyen „fejlesztői csapatban” dolgozni.

Nem utolsó sorban a projekt elkészítése által jobban megismerhettük a GitHub nyújtotta lehetőségeket.

## Források

- [1] - <https://www.szoftverpremium.hu/egyeb-microsoft-termek/microsoft-visual-studio/visual-studio-2022>
- [2] - [https://hu.wikipedia.org/wiki/C\\_Sharp](https://hu.wikipedia.org/wiki/C_Sharp)
- [3] - [https://hu.wikipedia.org/wiki/Modell-nézet-vezérlő](https://hu.wikipedia.org/wiki/Modell-n%C3%A9zet-vez%C3%A9rl%C3%B6)
- [4] - [https://archive.org/details/apacheseconomy00rist\\_938](https://archive.org/details/apacheseconomy00rist_938)
- [5] - <https://hu.wikipedia.org/wiki/REST>
- [6] - <https://azure.microsoft.com/hu-hu/overview/what-is-azure>
- [7] - <https://www.abplusz.hu>

Ikonok: <https://www.tutorialrepublic.com/bootstrap-icons-classes.php>

Bootstrap képek: <https://rentauto.hu>

# Ábrajegyzék

1. ábra: MNV Reprezentáció

(Forrás: <https://commons.wikimedia.org/w/index.php?curid=65388626>)

2. ábra: \_NavBar Partial View (Admin) (Forrás: saját képernyőfotó)

3. ábra: \_NavBar Partial View (Ügyfél) (Forrás: saját képernyőfotó)

4. ábra: \_Layout (Forrás: saját képernyőfotó)

5. ábra: Ügyfél model (Forrás: saját képernyőfotó)

6. ábra: Járművek model (Forrás: saját képernyőfotó)

7. ábra Szerződés model (Forrás: saját képernyőfotó)

8. ábra: Archivum model (Forrás: saját képernyőfotó)

9. ábra: Az Entity Framework (Forrás: <https://codewithmosh.com/p/asp-net-mvc>)

10. ábra: DbSet-ek (Forrás: saját képernyőfotó)

11. ábra: Adatbázistáblák (Forrás: saját képernyőfotó)

12. ábra: DbContext (Forrás: saját képernyőfotó)

13. ábra: Index Action (Forrás: saját képernyőfotó)

14. ábra: Új jármű és ügyfél hozzáadása (Forrás: saját képernyőfotó)

15. ábra: Járművek és ügyfelek szerkesztése (Forrás: saját képernyőfotó)

16. ábra: Mentés Action (Forrás: saját képernyőfotó)

17. ábra: Részletek Action (Forrás: saját képernyőfotó)

18. ábra: Formok (Forrás: saját képernyőfotó)

19. ábra: Ügyfél oldali validáció (Forrás: saját képernyőfotó)

20. ábra: AutoMapper (Forrás: saját képernyőfotó)

21. ábra: API GET (Forrás: saját képernyőfotó)

22. ábra: API GET (int id) (Forrás: saját képernyőfotó)

23. ábra: API POST (Forrás: saját képernyőfotó)

24. ábra: API PUT(int id) (Forrás: saját képernyőfotó)



- 25. ábra: API DELETE(int id) (Forrás: saját képernyőfotó)
- 26. ábra: jQuery-DataTable (Forrás: saját képernyőfotó)
- 27. ábra: Törlés script (Forrás: saját képernyőfotó)
- 28. ábra: Szerződés vezérlő, API DELETE (Forrás: saját képernyőfotó)
- 29. ábra: FilterConfig (Forrás: saját képernyőfotó)
- 30. ábra: Relációs adatmodell (Forrás: saját képernyőfotó)
- 31. ábra: Ügyfél törzs létrehozókód (Forrás: saját képernyőfotó)
- 32. ábra: Gépjármű törzs létrehozókód (Forrás: saját képernyőfotó)
- 33. ábra: Szerződések létrehozókód (Forrás: saját képernyőfotó)
- 34. ábra: Bejelentkezés try-catch (Forrás: saját képernyőfotó)
- 35. ábra: Bejelentkezés: foreach (Forrás: saját képernyőfotó)
- 36. ábra: Encrypt-Decrypt (Forrás: saját képernyőfotó)
- 37. ábra: ChildForm (Forrás: saját képernyőfotó)
- 38. ábra: ClickEvent (Forrás: saját képernyőfotó)
- 39. ábra: DGV (Forrás: saját képernyőfotó)
- 40. ábra: Kereső és szűrő feltétel (Forrás: saját képernyőfotó)
- 41. ábra: Input Form meghívása (Forrás: saját képernyőfotó)
- 42. ábra: Adat hozzáadása (Forrás: saját képernyőfotó)
- 43. ábra: Adat törlése (Forrás: saját képernyőfotó)