**Ceglédi SZC Mihály Dénes Szakképző Iskola**

**Szoftverfejlesztés és tesztelés szakirány**

**Esti Tagozat**

**Szakdolgozat**

**Autó Kölcsönző**

*CityCars*

**Készítette:**

*Balla Zoltán*

**Gyömrő**

**2022**

**Tartalomjegyzék**

[1. Bevezetés 3](#_Toc102834609)

[2. Fejlesztői dokumentáció 5](#_Toc102834610)

[2.1 Fejlesztői környezet 5](#_Toc102834611)

[2.2 Adat szerkezet bemutatása 5](#_Toc102834612)

[2.3 Frontend bemutatása 8](#_Toc102834613)

[2.4 Backend bemutatása 15](#_Toc102834614)

[2.5 Fejlesztési lehetőségek 18](#_Toc102834615)

[2.6 Teszt dokumentáció 19](#_Toc102834616)

[3. Felhasználói dokumentáció 21](#_Toc102834617)

[3.1 Szükséges hardver eszközök 21](#_Toc102834618)

[3.2 A program részletes bemutatása 21](#_Toc102834619)

[3.3 Kapcsolattartás 26](#_Toc102834620)

# **Bevezetés**

A szakdolgozatomra elkészített asztali programom, egy autókölcsönző cég alkalmazottainak készült, melyben a cégük személygépjárműveit tudják nyilván tartani, továbbá lehetőségük van különböző módosításokat végrehajtani.

Igyekeztem minél felhasználóbarátabb felületű programot készíteni, az iskolában tanult nyelv segítségével.

Az indokolta a téma választást, hogy közel állnak hozzám a gépjárművek szeretek velük kapcsolatban párbeszédbe elegyedni, és nem is olyan rég, ezelőtt 2-3 éve történt egy közúti balesetem ahol igénybe kellet vennem a szolgáltatást, ahol nagyon segítőkészek voltak az ott dolgozók, és a tanácsukkal sikerült autót bérbe vennem a biztosító kontójára, amíg az enyém a szervízben elkészül.

Volt más eset is amikor autót kellet bérelnem, egy külföldi utazás során. Ezelőtt 4-5 éve, ahol kezdetleg elég körülményesen indultak a bérbe vételi folyamatok, de a végére kitisztultak és sikeresen bérbe tudtam venni a kiszemelt gépjárművet.

Azt gondolom, hogy napjainkban is sokan élnek a szolgáltatást nyújtó cégek segítségével, hisz a mai felgyorsult világban, szinte minden háztartásban van egy személygépkocsi, amivel előfordulhatnak bizonyos gondok, vagy akár baleset is. Ilyenkor sok mindentől eleshet az illető, kimaradhat a munkából, amit nem biztos, hogy jó szemmel néznek majd a kollégái vagy éppen a vezetőség, esetleg nem tud elmenni a gyerekért az óvodába, iskolába vagy sürgős esetekben egy közeli családtagért a kórházba.

Ezekben az esetekben fordulunk az efféle finanszírozási formákhoz, sokak mindennapi életéhez elengedhetetlen kellék az autó, illetve külföldi nyaralásnál is kifejezetten nagy segítséget tudnak nyújtani a kölcsönző cégek.

# **Fejlesztői dokumentáció**

## **2.1 Fejlesztői környezet**

A program megírásához a c# nyelvet választottam, azért esett rá a választásom, mert ezt a nyelvet érzem magamhoz közelebbinek, és ezt a nyelvet használtuk leginkább az iskolában.

A fejlesztést a saját gépemen eszközöltem ami egy Acer típusú laptop.

Főbb tulajdonságai:

* Windows 10 Professional 64 bites verzió
* Intel Core i5 7200U 2,5Ghz
* 8 GB DDR4
* 1000 GB HDD
* 256 GB SSD
* Nvidia GeForce 940MX 2 GB videókártya
* 15,6 Coll monitor

A CityCars nevű program ezen a gépen könnyedén fut, a program futtatásához ennél elég egy kevésbé jól felszerelt hardver is. Ezt a gépet nem programozásra szereztem be még annak idején, de többé-kevésbé szépen teszi a dolgát.

A szoftver amit használtam a program létrehozásához az a Microsoft által kifejlesztett Microsoft Visual Studio azon belül is a 2022-es kiadása, azon belül is a Community verzió. A szoftver választásának főbb okai azok, hogy ezt használtuk az iskolai órákon, és ezt tudtam jobban elsajátítani, és így közelebb is éreztem magamhoz.

## **Adat szerkezet bemutatása**

Az elkészített program alapvetően kettő táblázatot használ egy adatbázison belül, amivel a felhasználó dolgozni tud. Benne vannak az alapvető Sql utasítások amiket a legsűrűbben használunk mint például a kiválasztás, a módosítás, a törlés, és az új elem hozzáadása. Ezeket az utasításokat az adatbázison keresztül használva juthatunk hozzá a kívánt módosítás eléréséhez. A program az összes input mezőre amit az esetek többségében textbox-al oldottam meg, kivétel kezelést alkalmaz az esetleges üres mezők, a félregépelések, vagy a rossz formátumban megadott beviteli érték elkerülése érdekében. Ez ugye bár egy alkalmazottak részére létrehozott kis program amiben kettő táblát láttam nagyon hasznosnak létrehozni az adatbázison belül aminek CityCars a fantázia neve. Az egyik tábla a személygépkocsikról szóló különböző adatokat tárolja.

Tulajdonságai szerint:

* Id
* Rendszám
* Évjárat
* Gyártmány
* Típus
* Motor mérete és üzemanyaga
* Színe
* Extrák tartalma
* Ára Ft-ban napra leosztva
* Hozzá adás dátuma

Ezeket a tulajdonságokat mindenféleképpen meg kell adni a táblázatban ezért amikor létre hoztam a táblát ezt a tulajdonságot be is állítottam a „NOT NULL” parancsokkal, hogy a felhasználó még véletlenül se hagyhatja ki.

* Az Id-nál int típusú változót használtam Identityvel, ami 1-esével növekszik automatikusan, és elsődleges kulcs szerepe van.
* A Rendszámnál Varchar típusú változót használtam maximális 7 karakterrel, hogy a magyar rendszámú gépkocsikat gond nélkül fel lehessen venni, ugyan is a cég nem dolgozik más országbeli gépkocsikkal. Így például a következő formátumú rendszám felvételére alkalmas az alkalmazás: ABC-123.
* Az Évjáratnál Int típusú változót használtam, mivel itt csak a gyártási év érdekel minket, mindenféle egyéb karakter nélkül.
* A Gyártmánynál Varchar típusú változót használtam maximális 20 karakterig terjedő érték megadással. Úgy gondolom, hogy ennek elégnek kell lennie, a tapasztalataim szerint.
* A Típusnál Varchar típusú változót használtam maximális 20 karakterig terjedő érték megadással. Úgy gondolom, hogy ennek elégnek kell lennie, a tapasztalataim szerint, mivel nem ismerek ennél hosszabb típus értékű gépjárművet.
* A Motornál szintén Varchar típust használtam itt 20 karakterig terjedő maximális érték megadásával. Itt a motor henger űrtartalmára vagyunk kíváncsiak és arra, hogy azt milyen üzemanyag hajtja. Ez ugyebár lehet benzines, dízeles, hibrides, vagy elektromos meghajtású is.
* A Színnél Varchar típusú változót használtam maximális 20 karakterig terjedő érték megadással. Úgy gondolom, hogy ennek az értéknek elégnek kell lennie.
* Az Extráknál Is Varchar típusú változót használtam itt már 50 karakterig terjedő maximális érték megadással. Itt alapvető extrákra vagyunk kíváncsiak, amiktől nőhet a kölcsönbe adni kívánt gépkocsi értéke, és amivel a bérlők vonzóbbnak találhatják a saját maguk részére, vagy éppen egy harmadik személynek akit meg akarnak vele lepni.
* Az Árnál Int típusú változót használtam, mivel itt megint csak számokat kérünk a felhasználótól, mivel ez a táblázatban és a formban egyértelműsítve lett, hogy Ft/nap-ban gondolkozzon az adat rögzítő kolléga.
* A Dátumnál Date típusú változót használtam, elkerülve azt, hogy kötelező legyen időt is megadni, mivel azt nem tartom lényeges információnak. Ennél a táblázatnál a dátumnál formátumi megkötést alkalmaztam „(yyyy-MM-dd)”, ami, hogyha nem teljesül akkor nem fog módosulni a táblánk tartalma, mivel más formátumot nem ismer fel. Ezt a bérlők táblánál már másképp oldottam meg. Egy datetimepicker segítségével könnyedén kiküszöbölhető ez a formátum rosszul használási hiba, itt azért hagytam ezt a módszert, hogy legyen egy ilyen verzió is, de bármikor át lehet programozni datetimepicker kompatibilisre ezt a táblát is.

Nos, így nézne ki a gépkocsik nyilvántartó táblája.

A másik tábla a bérlőkről szóló különböző adatokat tárolja.

Tulajdonságai szerint:

* Id
* Személyi igazolvány száma
* Neve
* Irányítószáma
* Városa
* Címe
* Hozzáadás dátuma

Ezeket a tulajdonságokat mindenféleképpen meg kell adni a táblázatban ezért amikor létre hoztam a táblát ezt a tulajdonságot be is állítottam a „NOT NULL” parancsokkal, hogy a felhasználó még véletlenül se hagyhatja ki.

* Az Id-nál int típusú változót használtam Identityvel, ami 1-esével növekszik automatikusan, és elsődleges kulcs szerepe van.
* A Személyi igazolvány számnál Varchar típusú változót használtam 8 karakteres maximális értékbeviteli lehetőséggel, mivel 6 számból és 2 betűből áll.
* A Névnél Varchar típusú változót használtam maximális 50 karakter beviteli lehetőséggel. Így ebbe számításaim szerint az is bele fog férni aki férjnél van, doktor vagy professzor vagy idősebb, ifjabb, vagy özvegy titulusban van.
* Az Irányítószámnál Int típusú változót használtam, mivel itt csak számokat fogunk használni.
* A Városnál Varchar típusú változót használtam 30 karakteres maximális karakter megadással, mivel Magyarországon nincsen hosszabb karakterű település.
* A Címnél Varchar típusú változót használtam 50 karakteres maximális karakter megadással, így biztosan elég lesz a leghosszabb utca nevek megjelenítésére is, itt az utca/út nevére vagyunk kíváncsiak egy házszámmal ellátva.
* A Dátumnál Date típusú változót használtam, elkerülve azt, hogy kötelező legyen időt is megadni, mivel azt nem tartom lényeges információnak. Itt datetimepickert használtam a következő rögzített formátummal: „(yyyy-MM-dd)”. Itt a datetimepicker segítségével kiküszöböltem, hogy az adat rögzítő véletlenül is akár vagy direkt rossz formátumot használjon, így itt nem törhet el a program.

## **Frontend bemutatása**

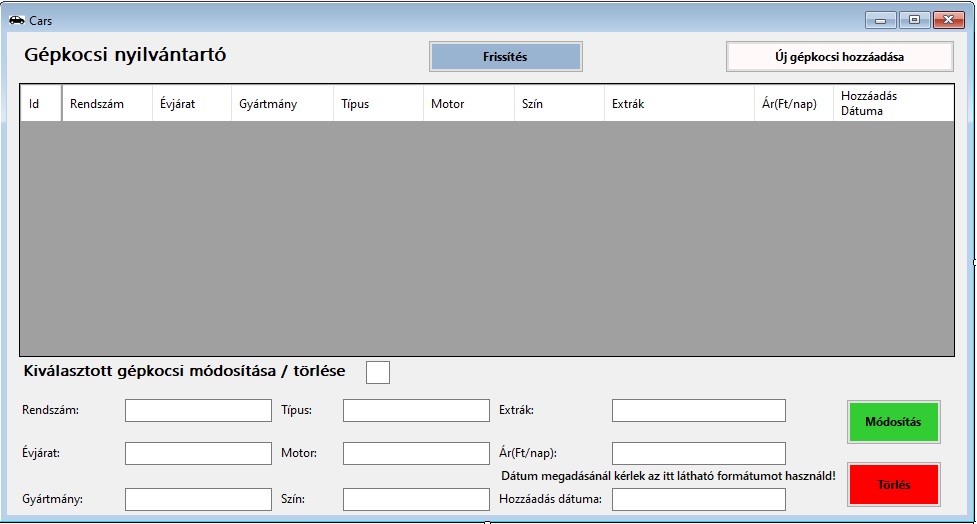
Az elkészült alkalmazásban több Form felületet is használva lett, mert szerettem volna, ha áttekinthetőbb, és falhasználóbarátabb az alkalmazás. Létrehoztam egy úgynevezett fő ablakot ahol választani lehet, hogy a személygépkocsiknak a nyilvántartásával szeretne foglalkozni az adat rögzítő vagy a bérlők nyilvántartásával. Itt egyszerre csak egy ablakban tud dolgozni az adatrögzítő, mivel úgy lett megoldva, hogy amit éppen megnyit ablakot, csak abban tud dolgozni, egy ShowDialog paranccsal. Az előző ablak attól függetlenül még él, és nincs bezáródva, csak éppen a belőle megnyitott ablak lesz az elsődleges. Miután bezártuk ezt az ablakot, visszatérünk ahhoz az ablakhoz amelyből megnyitottuk, és ismét dolgozhatunk benne.

*1.ábra*

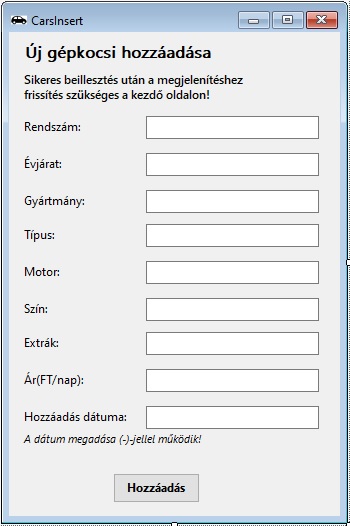
Az *„1.ábrán”,* látható a fő ablak, egy klasszikusnak mondható egyszerű kezelő szervekkel ellátva. Található rajta kettő gomb, amivel válthat az éppen foglalkozni vágyott felületre. A fejlécben a vállalkozás neve található, egyedi betűstílussal, alatta pedig az alkalmazás mivolta.

Ha a fő ablakból a *„Gépkocsik nyilvántartása”* gombra kattintunk, megjelenik egy másik felület, ahol a gépkocsik nyilvántartásában tudunk különböző módosításokat végezni.

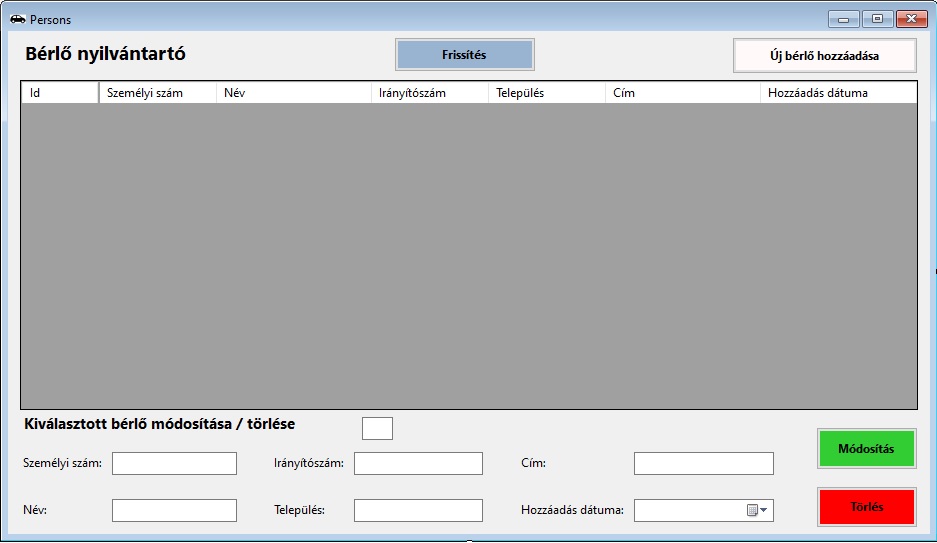
Ahogy a *„2.ábrán”* is látható, itt már egy komplexebb felülettel találkozunk. A form címét egy label-lal oldottam meg, egyénileg megformázva. Több gombot is találhatunk rajta.

* A *„frissítés”* gombnak kulcs szerepe van ezen az ablakon, kezdetkor ezzel fogja feltölteni a datagridview-t, a már meglévő adatokkal, ha ez a gombnyomás nem történik meg, akkor a datagridview nem fog megjeleníteni semmit. Ezt a gombot kell megnyomni akkor is, amikor új gépkocsit adunk a táblázatunkhoz, hogy frissüljön, különben a datagridview-n nem fog megjelenni. A többi esetben automatikusan lett megoldva a frissítést. Formázását, úgy próbáltam megoldani, hogy egyértelművé tegye a gomb szerepét.
* Az *„Új gépkocsi hozzáadása”,* gombot választva megjelenik egy teljesen új felület(*„3.ábra”*), amin új gépkocsit lehet majd beszúrni a táblázatba, erről majd a későbbiekben. Formázását, úgy próbáltam megoldani, hogy egyértelművé tegye a gomb szerepét.
* A *„módosítás”* gombnak a szerepe is kulcsfontosságú, miután a datagridview betöltött, és megjelenítette az általunk kívánt elemeket, ezzel a gombbal tudunk majd módosítani a már benne szereplő adatokon. Formázását, úgy próbáltam megoldani, hogy egyértelművé tegye a gomb szerepét.
* **Végül a *„törlés”* gombbal eltávolíthatjuk egy általunk kijelölt elemet a táblázatból. Amiről majd egy messagebox fog minket értesíteni a sikeres műveletről. Formázását, úgy próbáltam megoldani, hogy egyértelművé tegye a gomb szerepét.

*2.ábra*

Található a formon egy datagridview is aminek a segítségével fogom megjeleníteni a kívánt elemeket. Az oszlopait a táblázat tulajdonságai szerint állítottam be, es úgy formáztam meg, hogy az át látható legyen. Az Id kapta a legkisebb oszlop szélességet, mivel itt lesznek a legrövidebben megjelenített karakterek, és az Extrák nevű oszlop kapta a legszélesebb szélesség beállításokat, mivel itt számítunk majd a leghosszabban megjeleníteni kívánt karakterekre. A RowHeaders láthatóságát false értékre állítottam, hogy ezzel ne foglalja az így is kis méretű datagridview-on megjelenített elemek helyét. Ki lettek kapcsolva a különböző módosítási lehetőségek a felhasználó számára, mint például: módosítás, törlés, és beszúrás, így a datagridview-t csak fejlesztő tudja a továbbiakban módosítani. Az AutoSizeColumnsMode-ot Fill-re állítottam, és azon belül százalékos érték megadással módosítottam az oszlop szélességeket. A datagridview-on belül létrehoztam egy CellClik Event-et is aminek a segítségével az alatta található textbox-okat töltöm fel a label-el megjelölt és a datagridview oszlop nevének megegyezően tartalommal. Ezek a textbox-ok fognak majd segítségül szolgálni a módosításban, mivel akkor fog módosítást végre hajtani, ha ezekben a textbox-okban változás történt, és mindegyik textbox a formátumnak megfelelően ki van töltve, és nem lett üresen hagyva, különben figyelmeztető üzenetet kap az adatrögzítő. Miután módosítottunk az adatok automatikusan frissülnek. Itt létre hoztam egy textbox-ot az Id-nak is, mivel erre hivatkozok a különböző módosításoknál, aminek a láthatóságát false-ra állítottam, mert ezt a felhasználónak nem szükséges látnia. Ezeket a datagridview alatt elhelyezkedő elemeket elválasztottam egy alcímmel, amivel azt próbálom sugallani a felhasználó felé, hogy ott tud módosításokat eszközölni.

*3.ábra*

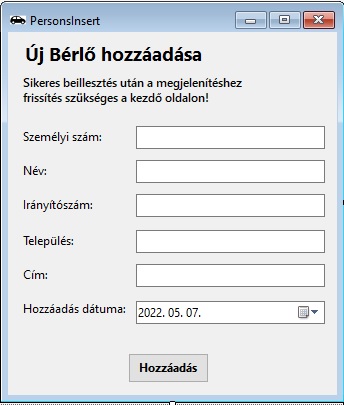
Az *„Új gépkocsi hozzáadása”* gomb megnyomásával egy új felületet kapunk *(„3.ábra”)*, aminek a segítségével új gépkocsit tudunk hozzáadni a táblázatunkhoz. A form címét egyedi formázással oldottam meg. textbox-okat, label-eket, és gombot használtam a megjelenítéshez, az szerint ahogy a táblázatban is lesznek feltöltve. A dátumnál itt textbox-ot használok, formátum megkötéssel, amit jelzek is a felhasználónak, ha nem így tölti ki az input mezőt, akkor nem fog megtörténni a beszúrás parancs, és nem megfelelő formátumra hivatkozva fogja figyelmeztetni az adat rögzítőt. Ha minden mezőt jól töltöttünk ki, akkor sikeresen beszúrja az új adatokat a táblázatba, amit frissítéssel tehetünk láthatóvá a fő ablakon.

*4.ábra*

Ha a fő form-ban a *„Bérlők nyilvántartása”* gombra kattintunk a *(„4.ábra”)* fog megjelenni előttünk. A form címét egy label-lal oldottam meg, egyénileg megformázva. Több gombot is találhatunk rajta.

* A *„frissítés”* gombnak kulcs szerepe van ezen az ablakon is, kezdetkor ezzel fogja feltölteni a datagridview-t, a már meglévő adatokkal, ha ez a gombnyomás nem történik meg, akkor a datagridview nem fog megjeleníteni semmit. Ezt a gombot kell megnyomni akkor is, amikor új bérlőt adunk a táblázatunkhoz, hogy frissüljön, különben a datagridview-n nem fog megjelenni. A többi esetben automatikusan oldottam meg a frissítést. Formázását, úgy próbáltam megoldani, hogy egyértelművé tegye a gomb szerepét.
* Az *„Új bérlő hozzáadása”,* gombot választva megjelenik egy teljesen új felület(*„5.ábra”*), amin új bérlőt lehet majd beszúrni a táblázatba, erről majd a későbbiekben. Formázását, úgy próbáltam megoldani, hogy egyértelművé tegye a gomb szerepét.
* A *„módosítás”* gombnak a szerepe is kulcsfontosságú, miután a datagridview betöltött, és megjelenítette az általunk kívánt elemeket, ezzel a gombbal tudunk majd módosítani a már benne szereplő adatokon a textboxok segítségével. Formázását, úgy próbáltam megoldani, hogy egyértelművé tegye a gomb szerepét.
* Végül a *„törlés”* gombbal eltávolíthatjuk egy általunk kijelölt elemet a táblázatból. Amiről majd egy messagebox fog minket értesíteni a sikeres műveletről. Formázását, úgy próbáltam megoldani, hogy egyértelművé tegye a gomb szerepét.

A formon belül elhelyeztünk egy datagridview-t is aminek a segítségével fogom majd megjeleníteni a Person táblában szereplő adatokat, a frissítés gombra kattintva. A datagridview AutoSizeColumnsMode-ját fill értékre állítottuk, így kitölti majd a fejlécet, oszlopait százalékosan állítottam be, a szerint, hogy melyik oszlop mekkora szélességet igényel. Ki lettek kapcsolva a különböző módosítási lehetőségek a felhasználó számára, mint például: módosítás, törlés, és beszúrás, így a datagridview-t csak fejlesztő tudja a továbbiakban módosítani. A RowHeaders értékét false-ra állítottam, hogy ne vegyen el felesleges helyet az így is kis méretű datagridview-ból. A form alján itt is label-ek és többnyire textbox-ok segítségével lett megoldva a megjelenítés, ami később szerepet játszik a különféle módosításokban. Ezeknek az input mezőknek a kitöltését datagridview ClickEvent-jének a segítségével fogjuk eszközölni. Itt lehet majd módosításokat is létrehozni, ha a megfelelő formátumban lettek megadva az adatok, különben minden egyes mezőre beállított hibaüzenet fog megjelenni, egészen addig, míg nem sikerül helyesen begépelni az adatot. A dátum label mellett itt datetimepicker található, a gépkocsi nyilvántartó formján megszokott textbox helyett, így kiküszöbölve az esetleges formai hibázásokat, elgépeléseket. A datetimepicker alapértelmezett értékét itt üresnek állítottuk be, és csak akkor jelenik meg benne érték, ha behívjuk a datagridview ClickEvent-jét, vagy manuálisan állítunk rajta. A dátum tartomány minimális értékét 1990.01.01-re lett beállítva, míg a maximális érték 2050.12.31-re lett módosítva, ezzel kiküszöbölve irreálís érték megadását. A formátuma egyénileg lett beállítva, short verziót kapott „(yyyy-DD-mm)” formátummal. A formon létre lett hozva egy textbox az Id-nak is, de csak a fejlesztés szempontjából, így ez a felhasználó elől el lett rejtve.

Az *„Új bérlő hozzáadása”* gombra kattintva, kapunk egy új formot *(„5.ábra”)* amit ShowDialog parancsal jelenítettünk meg, így az újonnan megnyílt ablak lesz az aktív ablak, és az anya ablakban nem tudunk dolgozni míg ezt be nem zártuk. Bezárás után viszont ujjra visszatérhetünk a kiindulási ponthoz. Az ablak címe label-lal lett megjelenítve, egyénileg formázva. Ha jobban megnézzük itt is label-ek és textbox-ok segítségével fogjuk felvinni az adatbázisunkba az új adatokat, kivéve a dátumnál. Ugyanis itt datetimepicker használatával fogjuk megadni az éppen aktuális rögzítési dátumot, amit alapértelmezetten a mai dátumra lett beállítva, hiszen valószínűsíthetjük, hogy akkor fog csatlakozni a bérlők közösségéhez, az új ügyfél. A minimális értékét 1990.01.01-re állítottuk be, míg a maximális értéke itt is a 2050.12.31-es dátum napját viseli, ezzel kiküszöbölve az irreális értékek megadását. Miután megfelelően kitöltöttük az input mezőket, rákattinthatunk a módosítás gombra, ami befogja illeszteni az újonnan felvett bérlő adatait a táblába, kivéve persze, ha nem sikerült minden adatot megfelelően kitölteni. Ugyanis itt kivétel kezelést alkalmaztunk, hogy kiszűrje az esetleges üres mezőket, és rossz formátum megadását. Ezt egy MessageBox-ban fogja látni a kedves felhasználó, ami segítséget nyújt majd neki a megfelelő kitöltéshez. Ha minden adat sikeresen ki lett töltve, akkor sikerült az adat beilleszttés, amit a fő ablakhoz vissza térve egy frissítésel tudjuk megjeleníteni.

*5.ábra*

## **Backend bemutatása**

A **mainFormon** kettő gombra kellet kódot írni, az egyikkel belépünk a gépkocsik nyilvántartásába, a másikkal a bérlők nyilvántartásába. Hasonlóan dolgozik mind a kettő gomb, kattintásra megjelenik az új form, és aktívvá teszi az éppen megnyíló formot, míg az anya ablak látható lesz ugyan, viszont inaktív.

A **gépkocsik nyilvántartásánál** egy kicsit több dolgunk volt ennél, először is, hogy kapcsolatba tudjunk lépni az adatbázisunkkal le kellet tölteni egy **NuGet Packages**-t, ami a **„System.Data.Sqlqlient”** névre hallgat. miután ez meg van a névtérben alkalmazni is kellett a **„using System.Data.Sqlqlient;”** névtérrel. Ezután kapcsolatba léphettünk a szerverrel egy **Sql Connection** változóval, ahol beállítottuk az elérési útvonalat.

Ezután létre kellet hozni egy metódust amivel lefut a feltöltés parancs, ez a metódus találóan a **„feltolt”** névre hallgat az alkalmazáson belül. Itt csatlakozni kellet az adatbázishoz, az elérési útvonal és az SqlConnection változó segítségével. Ezután az **open** paranccsal megnyitottuk a **kapcsolat**ot. Itt egy string típusú változóban el lett mentve az sql utasítás, az egyszerűség kedve miatt a későbbiekben így könnyebb lesz rá hivatkozni, amivel a táblázatban lévő összes adatot megfogja jeleníteni. Majd létre lett hozva egy **SqlCommand** nevű változó amiben meg lett adva ez a string utasítás, és az elérési útvonal string-je. Így létrejött a kapcsolat az adatbázisunkkal. Létrehoztunk egy **SqlDataReader** osztályt *„lekerdez”* néven, amivel kiolvastuk az adatokat a táblából egy **ExecuteReader** methodus segítségével, és egy **while ciklus**-al lefuttattuk a táblázatunkon addig amíg van olvasható érték a *„lekerdez.read”* értékkel. Előtte viszont töröltük a datagriedview sorait és az összes input mezőt, erre a lépésre azért van szükség, mert ha nem törlünk előtte, akkor duplikálva fognak megjelenni az adatok a datagriedview-ban. Mielőtt feltöltenénk a datagriedview sorait a while cikluson belül a megadott sorrendben a datagriedview nevével és a (.Rows.Add) paranccsal, még át kell konvertálni a dátumot a megfelelő formátumba, hogy már alapból így jelenítse meg a datagriedview-ban, és ezáltal a ClickEvent-en belül az input mezőben is. Ehhez létre kellet hozni egy DateTime nevű változót amit DateTime-vá konvertáltunk a sorrendben megfelelő *lekerdez* nevű változót ami a dátumot tárolja. Ezután string-é konvertáltuk egy megadott formátumban „(yyyy-MM-dd)”, és így lett elmentve egy string dt nevű változóban, amit később felhasználtunk a kiíratásnál. Itt ugye sorrendben feltöltöttük a sorokat és **zártuk a kapcsolatot**.

Létre kellet hozni egy olyan metódust is ami törli az input mezőket ez a metódus az **ures** nevet viseli, amikor ezt meghívjuk a formon belül törlődik minden input mező.

Ezután a **Frissít** gombra meghívtam a *feltolt* metódust, ami kattintásra lefut, és feltölti a datagriedview-t, a megfelelő sorrendben.

Itt kapott szerepet a datagriedview eventjein belül a CellClick event ami a datagriedview táblázatán belüli kattintás után feltölti a textbox-okat a megfelelő sorrendben. és értékkel, itt kellet egy olyan kikötést kötni, hogy a click a táblázaton belül legyen, ha például a ColumnHeaders-en történik ez az event, akkor egy ürítő metódus kap szerepet, ami üresre állítja a textboxokat. Viszont mielőtt ezeket az értékeket beillesztenénk a textbox-okba string típusúvá kell átalakítani. Innentől kezdve tudunk dolgozni az ott szereplő adatokkal.

Itt kap majd szerepet a **módosítás** gomb. A módosítás gombbal a textbox-okban lévő adatokat tudjuk módosítani. Itt is meg kellet nyitni hozzá a kapcsolatot az adatbázisunkal, majd kellet írni egy Sql utasítást, az **SqlDataAdapter** osztály segítségével. Majd az **UpdateCommand** segítségével eltárolunk egy utasítást amiben hivatkozunk az input mezőkre, és a ColumnHeaderek-re majd frissítjük az új értékekkel a már meglévő adatokat. Ezt egy **TRY**-ba lett megoldva, ahol szerepe van a **CATCH**-nek is, ugyanis nem megfelelő formátumnál, hibát fog kapni a felhasználó a segítségével. Ha mindent jól csináltunk sikeresen fejeződik be a módosítás, és bezárul a kapcsolat. Persze a click metódusára az összes inputjára kivétel kezelést alkalmaztam **IF**-ek segítségével, amiben azt szabom meg, hogy nem lehet null értéke az input mezőknek. Miután lefut a metódus, törli a textbox-okat. és automatikusan feltölti a datagriedview táblába és az adatbázisba, és kapunk egy üzenetet, hogy sikerült a feltöltés.

Van még ezen a felületen egy **törlés** gombunk is amire szintén kellet egy kis rövid metódust írni. Itt hasonlóképpen jártunk el mint a módosításnál, egy pár apró különbséggel. Itt már a **deletecommand**-ot használtuk a törléshez, és az sql utasításban az id-nak a textbox-át vizsgáltuk, mert ez alapján fogjuk törölni az adatbázisunkból is a vele megegyező id-val szereplő elemeket. Itt is üzenetet kapunk, ha valami félre sikeredett, hogy tudjuk orvosolni a problémát. Miután rányomtunk a gombra a program az Id-nak a textbox-át fogja vizsgálni, hogy nem-e üres, mert ha igen, akkor nem történik semmi, és vissza térünk egy returnal. Ha viszont van benne tényleges érték, akkor összeveti a táblázatban valóval, és törli a kijelölt elemet. Majd üríti az input mezőket, és frissíti a datagriedview táblánkat, majd kapunk egy üzenetet a parancs végrehajtásának sikerességéről.

Ha minden igaz akkor egy gomb maradt ki ezen a felületen, az pedig nem más mint az **Új gépkocsi hozzáadása**, ha ezt a gombot aktiváljuk egy kattintással, akkor egy új form felületet fogunk kapni, aminek segítségével, új gépkocsikat fogunk tudni hozzáadni a táblázatunkhoz, amennyiben megfelelő értékekkel töltjük ki az ott megjelent input mezőket.

Itt létre kellet hozni egy metódust ami a *„SetDefaultValuesP”* névre hallgat, ami alap értékekre állítja a textbox-okat, tehát jelen esetbe üresre, ez minden kattintás után le fog futni, amennyiben helyesen töltöttük ki a mezőket.

Majd magának a *„Hozzáadás”* gombnak következett a parancs megfogalmazása, amit úgy kezdtünk, hogy létre kellet hozni a kapcsolatot az adatbázissal, a szerver nevére és a táblázat nevére hivatkozva. Itt megnyitottuk ezek után a kapcsolatot, és az **InsertCommand**-al létrehotunk egy Sql utasítást, ami a táblázatnak adja az input mezőben lévő értékeket, abban a sorrendben, amiben megadjuk neki. Itt a szám formátumú beviteli értékeket át kellet konvertálni a sikeres lefutáshoz **Int.Parse**-val. Ha ez valamilyen oknál fogva nem sikerült a catch segítségével megállítjuk a program futást és egy hibaüzenetet dobunk ki az adat rögzítőnek, hogy segítségével újra megpróbáljon helyes adatokat rögzíteni a táblánkba. Ha ez sikerült zárjuk a kapcsolatot. Miután megnyomtuk **a Hozzáadás** gombot az éppen aktuális formunkon, a rá megírt metódus ellenőrzi if-ek segítségével, hogy nem-e lett üresen hagyva véletlenül valamelyik mező, ha igen, akkor figyelmeztet a mezőre hivatkozó tulajdonságokkal, hogy a felhasználó könnyedén megtalálja, hogy mi az amit nem töltött ki. Ha mindent sikerült megfelelően kitölteni akkor, a gépkocsik nyilvántartása felületen, szükséges egy frissítés gombot nyomnunk, a teljes sikerért, és így már valóban látható lesz az eredmény. Itt felfogja tölteni az új adatokkal a táblánkat ez a gombnyomás, és alapértelmezett értékekre állítja az input mezőket.

Ezek után, ha ezeket az ablakokat bezártuk, és vissza térünk a MainForm-hoz, ott választhatjuk a **Bérlők nyilvántartása** lehetőséget gomb formájában. Az itt megnyíló felületek is hasonlóképpen működnek, mint a fentebb már leírtak ezért nem is ismételném meg őket, csak átfutás képpen. Itt is a frissítés gomb megnyomása után tudunk majd bármit is tenni az alkalmazáson belül, hiszen ez fogja feltölteni adatokkal a datagriedview táblánkat, annyi különbséggel, hogy itt bérlőkkel fogunk dolgozni, és nem személygépkocsikkal tehát más adatok lesznek a datagriedview-ba. Ami leginkább változott az előzőekhez képest az a dátum megjelenítése, mivel itt nem textbox-ot használtunk, hanem datetimepicker-t ez egy sokkal letisztultabb, precízebb, és modernebb megjelenítést biztosít a felhasználó számára. Egyúttal a hibázási lehetőséget is leredukáltuk szinte nullára, mert nem tudja elgépelni a formátumot, hiszen megadott formátumban dolgozik, ami azért igényelt némi feladatot számunkra. Első körben itt is létre kellett, hozni egy datetime típusú változót amibe átkonvertáltuk a lekért adatot datetime típusúvá, majd elmentettük egy stringbe a megadott formátumon (yyyy-MM-dd), majd csak ezután irattuk ki a datagriedview-ra. Alapértelmezett értéknek itt is az üres értéket kapta, hiszen addig ne mutasson nekünk semmit, míg nem történt meg a ClickEvent és nem lettek feltöltve az input mezők. Miután megtörtént a ClickEvent itt is át kellet konvertáljuk a datagriedview-ba szereplő dátum értéket DateTime-ra, majd így adtuk oda a datetimepicker-nek, így hasonlóképpen tudta megjeleníteni mint a datagriedview-ba. A módosítás egy lenyíló ablakon történik, ahol tetszőlegesen ki tudjuk választani a megfelelő dátumot. Mielőtt módosítanánk itt is konvertálni kellett a datetimpepicker-ben lévő értéket datetime típusúvá majd egy stringbe elmenteni a már sokszor említett megadott formátumon, majd ezt oda adni az UpdateCommand Sql utasításnak.

A törlésnél változatlanul az Id-t ellenőrizzük, hogy nem-e mutat null értéket, ha nem akkor sikeresen töröltük a kijelölt elemet, amit egy messagebox is a tudtunkra fog adni.

Az **Új bérlő hozzáadása** gomb megnyomása után viszont valamivel másapp felületet kapunk, ugyanis más adatokkal dolgozunk, viszont lényegi eltérés nem igazán mutatkozik, kivéve a datetimepicker-rel való munkánk során. Itt alap értéknek az éppen aktuális dátumot fogja megkapni, hiszen nagy valószínűséggel ezen a napon fogják felvenni az új bérlőt a nyilvántartásba. Ahhoz, hogy sikeresen tudjuk beilleszteni az új adatokat a táblázatba, itt is meg kellett hívni a valuechanged eventet, ahol megadtuk neki a formátumát, amin szeretnénk, hogy megjelenítse az értékeket, majd beillesztés előtt át kellet konvertálni a már megszokott módon. Itt is kapunk hiba üzenetet, ha valamit rosszul gépeltünk be, vagy ha valamelyik input mező üresen maradt. Ha pedig sikeresen felvittük a kívánt adatokat, akkor már csak egy frissítés választ el minket attól, hogy ezt a datagridview-on is láthassuk. Ha pedig már nem használjuk az alkalmazásunkat akkor csak egyszerűen bezárjuk a bal felső sarokban található (X) gombal.

## **Fejlesztési lehetőségek**

A programot természetesen többféle módon lehetne még tovább fejleszteni, az egyik elgondolás az lenne, hogy kapna még egy gombot egy combobox-al, előszőr a gépkocsik nyilvántartása felületen, ahol gombnyomásra kiszűri a combobox-ban lenyíló lehetőségek közül azt a gépkocsi márkát amit éppen keresünk, másodszor pedig a bérlők nyilvántartásában ugyan ezen az elgondoláson kapná ezeket az elemeket, csak itt irányító számokra lehetne rá szűrni. Ez megkönnyítené a felhasználó dolgát, ha éppen egy bizonyos gépkocsit, vagy éppen személyt keres az adatbázisban. Persze ez a keresés szűrhet más tulajdonságra is tetszőlegesen. Ezt a **SELECT** **FROM** és a **WHERE** sql utasításokkal oldanám meg, ahol a combobox értékére hivatkoznék keresésnél, és így jeleníteném meg a datagriedview-on. Persze kapna egy label-t is ahol tájékoztatjuk a felhasználót, hogy éppen mit fog csinálni az alkalmazás, például „szűrés a következő gépjármű gyártmányra” itt lenne a combobox és utána kapna egy gombot amin a „szűrés” felirat szerepelne.

Tovább lehetne fejleszteni úgy is, hogy a gépkocsik nyilvántartásában létrehozni egy oszlop tulajdonságot ami a gépkocsik műszaki érvényességét tárolná dátum formátumba, így nyomon lehetne követni azt is, hogy meddig van műszaki az adott gépjárművön, és időben el lehetne vinni műszakiztatni még a határidő lejárta előtt.

## **2.6 Teszt dokumentáció**

Az elkészült alkalmazás többféleképpen lett tesztelve, hogy a szoftverben lévő hibákat még azelőtt megtaláljuk mielőtt, azt a végleges felhasználónak át adnánk. Folyamatosan tesztelés alatt volt a program, a megírása alatt is, hogy még idejében észre vegyük a szoftver által generált apróbb vagy nagyobb hibákat. Leginkább fejlesztői teszteléseknek esett alá a program az úgynevezett **fehér dobozos tesztelés**nek, ahol ugyebár ismerjük a termék forrás kódját.

Fehérdobozos tesztelés következtében a következő kipróbálási stratégiák érvényesültek:

* utasítás lefedés: minden utasítást legalább egyszer végre hajtottunk
* feltétel lefedés: minden feltétel legalább egyszer igaz, illetve hamis volt
* részfeltétel lefedés: Minden részfeltétel legalább egyszer igaz, illetve hamis volt

Leginkább statikus tesztelések zajlottak le a program fejlesztése közben, ilyenkor manuálisan lett átnézve a forráskód, és fejben lett lefuttatva, közben pedig gyanús elemeket kerestünk benne.

Itt két féle hibára szűrtünk leginkább, az egyik a **szintaktikai** ez ugyebár a fordítási hiba ahol elgépelések lehetnek például lemarad egy pontos vessző, vagy egy betű vagy ami még elég sűrűn elő szokott fordulni az az, hogy több esetleg kevesebb zárójelt rakunk oda ahova kell. Ilyenkor szintaktikai hibára hivatkozva nem fog lefutni a programunk. Sok esetben viszont szerencsénk van, mert a hibaüzenetben azt is kiolvashatjuk, hogy hányadik sornál akadt el a programunk futás közben. Így egyszerűen kikereshetjük, és megoldhatjuk a problémát.

A másik fajta hiba amit szűrtünk tesztelés során az pedig a **szemantikai** ez ugyebár a logikai hibára utal. Itt elégsokrétű volt a keresési folyamat.

* Kerestünk itt inicializálatlan változókat a forráskódban, hogy ilyenek ne fordulhassanak elő.
* Kerestünk itt értéknélküli függvényeket, hogy feleslegesen ne legyen rájuk hivatkozva.
* Kerestünk itt felhasználatlan értékeket, hogy feleslegesen ne generáljuk le őket.
* Kerestünk itt azonosan igaz elágazás feltételt.
* Kerestünk itt végtelen számlálós ciklust.
* És végtelen feltételes ciklust.

De természetesen volt dinamikus tesztelés is amikor a program futása közben teszteltük a programot. Itt segítségemre volt egy közeli ismerősöm is akinek úgy tudtam oda adni tesztelésre a programot, hogy nem volt programozói múltja. Ez az úgynevezett **fekete dobozos teszt**. Itt ugyanis az illető úgy tesztelte a programot, hogy a forráskód az nem volt elérhető számára, ezért csak egy felhasználói tesztet tudott végrehajtani. Ennél a fajta tesztelésnél több különféle adattal teszteltük a programot futása közben, helyessel és rosszall egyaránt, hogy minél több hibára fény derüljön. Sok esetben jó ha egy hozzá nem értő személy teszteli a programot, mert olyan hibákra is fényt tud deríteni, melyre egy tesztelő nem jönne rá, mert másképp gondolkozik. A tesztelés hasonló tulajdonságokkal bíró környezetben lett végre hajtva mint a fejlesztői környezet.

De természetesen **szürkedobozos tesztelés**nek is alá lett vetve. Itt a fekete és a fehér dobozos megközelítések egyszerre kerültek alkalmazásra. Ebben a megközelítésben ismertük a forráskódot, de viszont futtatásközben teszteltük a programot, így ha futásközben hibát észleltünk azt tudtuk javítani a forráskódba is.

# **Felhasználói dokumentáció**

Ez egy számítógépes szoftver, az autókölcsönző cég alkalmazottainak lett fejlesztve. Kizárólag asztali számítógépen vagy laptopon használható alkalmazás. A célja az, hogy az adatok rögzítését egyszerűbbé, és átláthatóbbá tegye, ezzel megkönnyítve a felhasználó adatokkal való foglalkozásait.

## **3.1 Szükséges hardver eszközök**

Ahhoz hogy használni tudjuk a szoftvert szükségesek hozzá minimális rendszer követelmények. Ezeket sorolnám fel pár sorban.

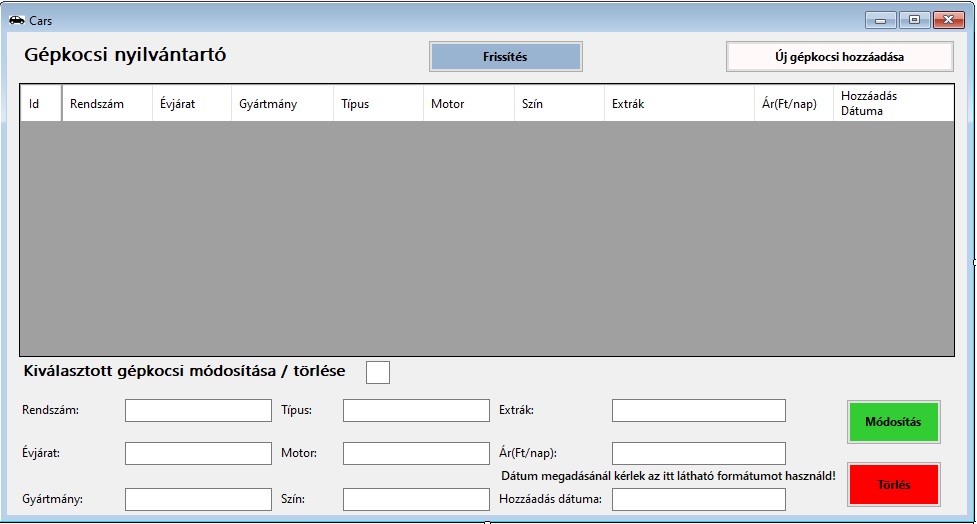
* Asztali számítógép, vagy laptop
* 1,8 Ghz vagy gyorsabb processzor
* 4magos vagy jobb processzor
* Windows 7 operációs rendszer vagy újabb
* Legalább 2GB Ram
* A merevlemezen legalább 200MB szabad hely
* Videokártya, amely támogatja a 720p (1280 x 720) minimális megjelenítési felbontást

## **A program részletes bemutatása**

Miután elindítottuk az alkalmazást egy kezdő ablak fog minket fogadni *(„1.ábra")* ahol láthatjuk az autókölcsönző cégünk nevét, és azt, hogy ez egy nyilvántartó rendszer az alkalmazottak részére. Ezen az oldalon kettő gomb fog minket fogadni az egyik a **Gépkocsik nyilvántartása**, ha ezt a lehetőséget választjuk akkor a gépkocsik nyilvántartásával fogunk tudni dolgozni, a másik a **Bérlők nyilvántartása**, ha ezt a lehetőséget választjuk akkor a bérlők nyilvántartásával fogunk tudni dolgozni. Azt tudni kell, hogy ha bezárjuk az ebből megnyíló ablakokat, akkor ide a fő ablakhoz fogunk mindig vissza lyukadni. Egészen addig, míg be nem zárjuk ezt az ablakot is, ugyan is akkor a program is bezárul.

*1.ábra*

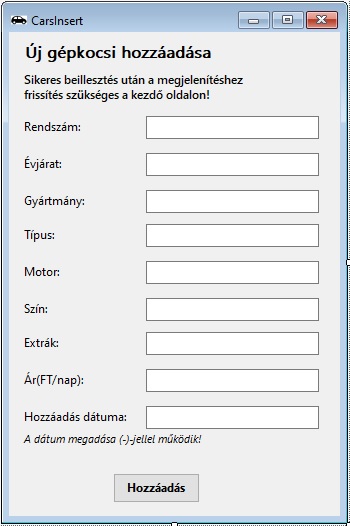
Miután kiválasztottuk a gépkocsik nyilvántartása lehetőséget, megjelenik a gépkocsi nyilvántartó ablakunk *(„2.ábra”)*, ahol különböző módosításokat tudunk végbe hajtani a gépkocsik adatbázisában. Az első lépés hogy frissítsük a felületet, ez a **Frissítés** gombbal lehetséges, ekkor betöltődik a tábla, és megjelennek az adatok az ablak közepén található táblázatban. Ezután, ha a táblázat valamely sorára kattintunk ahol adat is található, azt megjeleníti nekünk a táblázat alatt található szöveg dobozokban. Azért fontos, hogy oda kattintsunk ahol adat is található, mert ha olyan részre kattintunk, ahol ez nem érvényesül a szövegdobozok nem változnak, vagy üressé válnak. Miután betöltöttük az elemeket a szövegdobozokba, azután a tőlük jobb oldalra található gombokkal tudunk tovább dolgozni az adatokkal.

*2.ábra*

Az első gomb amit láthatunk fentről lefelé az a **Módosítás**, ezzel a gombbal a szövegdobozok értékét tudjuk módosítani, és ha megfelelő formátumban módosítottuk az adatokat a módosítás gombra kattintva megváltozik arra az értékre amelyre átírtuk. Ha nem sikerült a megfelelő formátumot használnunk, akkor hiba üzenetet kapunk, amely segítségével újra próbálkozhatunk. Arra még nagyon figyelni kell, hogy ne hagyjunk üres mezőket, mert arra allergiás a program, ezt hiba üzenettel jelezni is fogja felénk.

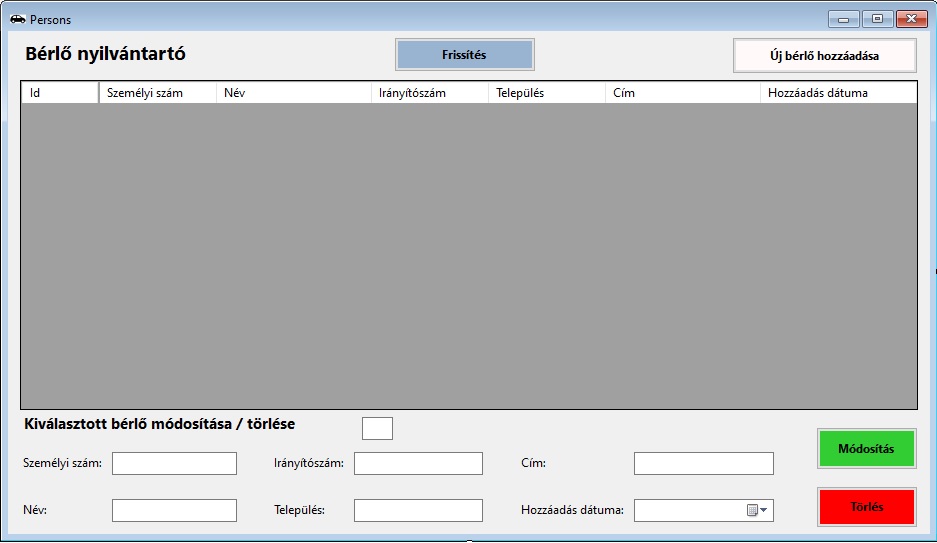
A következő gomb a **Törlés** lesz, itt a kijelölt elemet véglegesen eltávolítja az adatbázisunkból, amely végén egy üzenetet is olvashatunk, hogy sikeres volt az eltávolítás.

Ami még kimaradt az a jobb oldalt felül található **Új gépkocsi hozzáadása** gomb, mellyel új gépkocsikat adhatunk hozzá az adatbázisunkhoz. A gomb megnyomásával egy új felületet kapunk („3.ábra”) amely segítségével ezt végre is vihetjük. Itt is szöveges dobozokat találhatunk, amely előtt szerepel a szövegdoboz címe, amely szerint fel kell tölteni a különböző tulajdonságait a személygépkocsinak. A dátumnál figyeljünk a megfelelő formátum bevitelére, amelyre egy kis szöveg is figyelmeztet minket. Itt is figyelni kell arra, hogy az összes mező ki legyen töltve, mert különben sikertelen lesz az adat beillesztés.

*3.ábra*

Ha mindent megfelelően a mező címek szerint kitöltöttünk, akkor rányomhatunk a **Hozzáadás** gombra, amely használatával a szövegdobozokban lévő adatok hozzá adódnak a már meglévőkhöz. Ha nem kapunk hiba üzenetet sikeres volt az adatbeillesztés, megjelenítéséhez, viszont egy frissítés szükséges a fő oldalon.

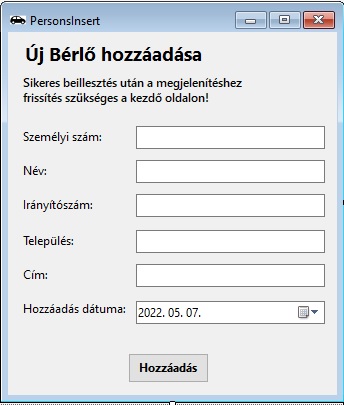
Vissza lépve a fő ablakhoz, ha a **Bérlők nyilvántartása** gombra kattintunk, akkor megjelenik a bérlők nyilvántartása ablakunk *(„4.ábra”).* Ahol különböző módosításokat tudunk végbe hajtani a bérlők adatbázisában. Az első lépés hogy frissítsük a felületet, ez a **Frissítés** gombbal lehetséges, ekkor betöltődik a tábla, és megjelennek az adatok az ablak közepén található táblázatban. Ezután, ha a táblázat valamely sorára kattintunk ahol adat is található, azt megjeleníti nekünk a táblázat alatt található szöveg dobozokban. Azért fontos, hogy oda kattintsunk ahol adat is található, mert ha olyan részre kattintunk, ahol ez nem érvényesül a szövegdobozok nem változnak, vagy üressé válnak. Miután betöltöttük az elemeket a szövegdobozokba, azután a tőlük jobb oldalra található gombokkal tudunk tovább dolgozni az adatokkal.

*4.ábra*

Az első gomb amit láthatunk fentről lefelé az a **Módosítás**, ezzel a gombbal a szövegdobozok értékét tudjuk módosítani, és ha megfelelő formátumban módosítottuk az adatokat a módosítás gombra kattintva megváltozik arra az értékre amelyre átírtuk. Ha nem sikerült a megfelelő formátumot használnunk, akkor hiba üzenetet kapunk, amely segítségével újra próbálkozhatunk. Arra még nagyon figyelni kell, hogy ne hagyjunk üres mezőket, mert arra allergiás a program, ezt hiba üzenettel jelezni is fogja felénk. Itt a dátumnál nem szövegdobozba fogjuk majd beírni az adatokat, hanem egy lenyíló naptár lesz a segítségünkre, ezáltal sokkal kevesebb esélyt adva rá, hogy hibázunk. Itt csak le kell nyitni a naptárat a naptár ikonra kattintva, és kiválasztani a hozzá adás dátumát. ha ez megtörténik, akkor automatikusan beilleszti az értékeket a felületre.

A következő gomb a **Törlés** lesz, itt a kijelölt elemet véglegesen eltávolítja az adatbázisunkból, amely végén egy üzenetet is olvashatunk, hogy sikeres volt az eltávolítás.

Ami még kimaradt az a jobb oldalt felül található **Új bérlő hozzáadása** gomb, mellyel új bérlőket adhatunk hozzá az adatbázisunkhoz. A gomb megnyomásával egy új felületet kapunk („3.ábra”) amely segítségével ezt végre is vihetjük. Itt is szövegesdobozokat találhatunk, amely előtt szerepel a szövegdoboz címe, amely szerint fel kell tölteni a különböző adatait az új bérlőnek.

*5.ábra*

A dátumnál itt is a már előző oldalon megszokott felületünket találhatjuk, itt alapértelmezettnek az aktuális dátum lesz megadva, mivel a legtöbb esetben aznap kerül felvételre az új bérlő, így sokszor ezzel a mezővel nem is kell foglalkoznunk. Itt is figyelni kell arra, hogy az összes mező ki legyen töltve, mert különben sikertelen lesz az adat beillesztés. Ha mindent megfelelően a mező címek szerint kitöltöttünk, akkor rányomhatunk a **Hozzáadás** gombra, amely használatával a szövegdobozokban lévő adatok hozzá adódnak a már meglévőkhöz. Ha nem kapunk hiba üzenetet sikeres volt az adatbeillesztés, megjelenítéséhez, viszont egy frissítés szükséges a fő oldalon.

Ha mind ezekkel megvagyunk, sikeresen megismerkedtünk az alkalmazással, és bezárhatjuk a programot.

## **Kapcsolattartás**

Ha bármilyen fejlesztéssel kapcsolatos kérdése van, észrevétele kérem keressen a [b.zolee6987@gmail.com](mailto:b.zolee6987@gmail.com) címemen.