**Szakképesítés neve:** **737/19T Szoftverfejlesztő képzés**

**OKJ száma:** **54 213 05**

SZAKDOLGOZAT

IMM (INDUSTRY MANUFACTURING MANAGER)

(Fejlesztői dokumentáció)

**Hidvégi Anikó Zeke Róbert**

Témavezető Tanuló

**Győr, 2021.**

Tartalom

[Témaválasztás indoklása 4](#_Toc71558994)

[Adatbázis kapcsolat és bejelentkezés 4](#_Toc71558995)

[Az adatbázis táblái 5](#_Toc71558996)

[(2. Kép. Adatbázis táblái) 5](#_Toc71558997)

[Program felépítése (tabPaged, mdiChild) 8](#_Toc71558998)

[Osztályok és metódusok 9](#_Toc71558999)

[Megjelenítőeszközök ( GridView, TreeView, ComboBox) 11](#_Toc71559000)

[GridView 11](#_Toc71559001)

[TreeView Találhatóak TreeView-ek a programban. Például a Termék Kategóriák menüpont alatt. Az említett TreeView így néz ki: 13](#_Toc71559002)

[ComboBox 14](#_Toc71559003)

[Frissítések 15](#_Toc71559004)

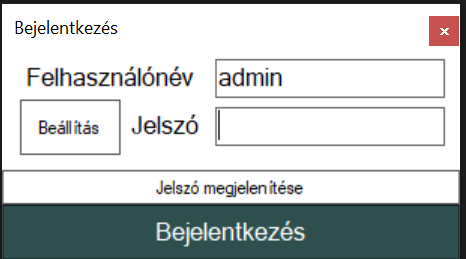
[Tovább fejlesztés és ötletek 15](#_Toc71559005)

## Témaválasztás indoklása

Szerettem volna egy olyan vállalatirányítási rendszert megalkotni, amely figyelembe veszi a felhasználói igényeket. Amelyben egy apró módosításért nem kell a program frissítésére várni, hanem azonnal, akár egy gombnyomással meg lehet oldali. Az ihletet a munkahelyemen használt MES rendszerből szereztem. Összegyűjtöttem az általam ismert operátorok véleményét és próbáltam belefejleszteni az IMM-be. A szoftver fejlesztéséből időhiányban nem megvalósítható hiányosságok, ötletek a „Tovább fejlesztés és ötletek” fejezetben kerülnek kifejtésre. A programot folyamatosan fejlesztem és a fejlesztés követhető és letölthető github-ról.

## Adatbázis kapcsolat és bejelentkezés

A program indulásakor a bejelentkező képernyő fogad minket. Ezen a képernyőn lehet beállítani az adatbázis kapcsolatot is.

  
*(1. kép. Bejelentkező ablak)*

A kapcsolat beállítását a „Beállítás” gombra kattintva tehetjük meg. Ezután az alap csatlakozáshoz szükséges paramétert kell megadni, jelenleg egy „Alapértelmezett” gombot lenyomva az alap adatbázis kapcsolati stringet fogja beírni. A mentés gombra kattintva elmenthetjük a beírt csatlakozási stringet a program beállításaiba. Ezek a beállítások felhasználókra vannak szabva, tehát ha egy másik eszközön is futtatni szeretnénk a programot, ezt ott is kötelező elvégezni. Az alap csatlakozási fájl az program indító fájlal egy mappában található „imm.db” néven. A forrásban létrehoztam egy Database osztályt ami a későbbiekben kezeli az adatbázis kapcsolatát.

public class Database {

public static string Connection = Properties.Settings.Default.dbConn;

public static void checkConnectionState(SqlConnection con) {

if (con.State == System.Data.ConnectionState.Open) {

con.Close();

}

}

}

Ennek az osztálynak nincsen konstruktora. Úgy szerettem volna megcsinálni, hogy példányosítás nélkül is ki tudjuk olvasni éppen ezért publikus statikus stringet és metódust hoztam létre. A string felelős az elmentett csatlakozási paraméter kiolvasására. A metódus pedig ellenőrzi, hogy a megadott „SqlConnection” állapota nyitva, ha pedig nyitva van, akkor bezárja a kapcsolatot.

## Az adatbázis táblái

## (2. Kép. Adatbázis táblái)

Használatban lévő táblák:

|  |  |
| --- | --- |
| FelhasznaloJogkorok | |
| fjid | INTEGER[PK] |
| fjmegnevezes | VARCHAR |
| fjerosseg | INTEGER |

Az „fjerosseg” oszlop jelenleg nincs használatban.   
Illetve a későbbi fejlesztések során át lesz dolgozva.

|  |  |
| --- | --- |
| Felhasznalok | |
| id | INTEGER[PK] |
| felhasznaloNev | VARCHAR(20) |
| jelszo | VARCHAR(20) |
| email | VARCHAR(50) |
| vezetekNev | VARCHAR |
| keresztNev | VARCHAR |
| jogkor | INT |

A „jogkor” oszlop jelenleg nincs használatban.

|  |  |
| --- | --- |
| GepAdatok | |
| id | INTEGER[PK] |
| gepid | INTEGER |
| geplokacio | INTEGER |
| gepmarka | VARCHAR |
| ciklusido | VARCHAR |
| karbantartaskezdes | VARCHAR |
| karbantartasismetlodes | VARCHAR |

|  |  |
| --- | --- |
| GepAllomasok | |
| gaid | INTEGER[PK] |
| gepid | INTEGER |
| allomasszam | INTEGER |
| allomasnev | VARCHAR |

|  |  |
| --- | --- |
| GepAllomasParameterek | |
| gaid | INTEGER[PK] |
| gepallomasid | INTEGER |
| parameternev | VARCHAR |
| parameterertek | VARCHAR |
| modositasdatuma | VARCHAR |
| modositotta | VARCHAR |

|  |  |
| --- | --- |
| Gepek | |
| id | INTEGER[PK] |
| gepnev | INTEGER |
| aktiv | VARCHAR |

|  |  |
| --- | --- |
| GepKategoriak | |
| id | INTEGER[PK] |
| gepid | INTEGER |
| kategoriaid | INTEGER |

|  |  |
| --- | --- |
| Kategoriak | |
| id | INTEGER[PK] |
| kategoriaNev | VARCHAR |
| aktiv | INTEGER |

|  |  |
| --- | --- |
| Lokaciok | |
| id | INTEGER[PK] |
| lokacionev | VARCHAR |

|  |  |
| --- | --- |
| MunkarendCsomagoals | |
| mcsid | INTEGER[PK] |
| mcstid | INTEGER |
| mcsdobozszam | INTEGER |
| mcscsomagolta | VARCHAR |
| mcsdatum | VARCHAR |

|  |  |
| --- | --- |
| MunkarendCsomagolas | |
| mcsid | INTEGER[PK] |
| mcstid | INTEGER |
| mcsdobozszam | INTEGER |
| mcscsomagolta | VARCHAR |
| mcsdatum | VARCHAR |

|  |  |
| --- | --- |
| Munkarendek | |
| mrid | INTEGER[PK] |
| mrazonosito | VARCHAR |
| mrmegnevezes | VARCHAR |
| mrdatum | VARCHAR |
| mrelkeszultdatum | VARCHAR |
| letrehozta | VARCHAR |

|  |  |
| --- | --- |
| MunkarendStatuszok | |
| mrsid | INTEGER[PK] |
| megnevezes | VARCHAR |
| szin | VARCHAR |
| sorszam | INTEGER |

|  |  |
| --- | --- |
| MunkarendTermekCsomagolas | |
| mtcsid | INTEGER[PK] |
| munkarendid | INTEGER |
| termekid | INTEGER |
| csomagolasdarab | INTEGER |
| csomagolaskezdete | VARCHAR |
| csomagolasvege | VARCHAR |
| csomagolta | VARCHAR |

|  |  |
| --- | --- |
| MunkarendTermekCsomagolas | |
| mtcsid | INTEGER[PK] |
| munkarendid | INTEGER |
| termekid | INTEGER |
| csomagolasdarab | INTEGER |
| csomagolaskezdete | VARCHAR |
| csomagolasvege | VARCHAR |
| csomagolta | VARCHAR |

|  |  |
| --- | --- |
| MunkarendTermekek | |
| mrtid | INTEGER[PK] |
| mrid | INTEGER |
| termekid | INTEGER |
| szuksegesdarabszam | INTEGER |
| statusz | VARCHAR |

|  |  |
| --- | --- |
| Raktarak | |
| rid | INTEGER[PK] |
| rmegnevezes | VARCHAR |

|  |  |
| --- | --- |
| RaktarakAlLokaciok | |
| ralid | INTEGER[PK] |
| rlid | INTEGER |
| allokacionev | VARCHAR |

|  |  |
| --- | --- |
| RaktarLokaciok | |
| rlid | INTEGER[PK] |
| rid | INTEGER |
| lokacionev | VARCHAR |

|  |  |
| --- | --- |
| Termekek | |
| id | INTEGER[PK] |
| termekNev | VARCHAR |
| termekNev2 | VARCHAR |
| cikkszam | VARCHAR |
| kategoria | INTEGER |
| csomagolasidarabszam | INTEGER |
| minimumgyarthato | INTEGER |
| raktar | INTEGER |
| aktiv | INTEGER |

## Program felépítése (tabPaged, mdiChild)

A program felépítése dockedTabpage. Azaz minden megnyitott modul dokkolásra kerül a main Formra. Ha egy adott modul már meg van nyitva és még egyszer megnyitom, akkor nem adódik hozzá a megnyitott oldalakhoz, helyette fókuszálva lesz és az lesz a kiválasztott TabPage. A fő eljárás a Main Form forrásában található.

void mdiMutat(Form \_frm) {

TabPage tp = new TabPage();

if (!formActive(\_frm)) {

tp.Size = this.Size;

\_frm.Dock = DockStyle.Fill;

\_frm.TopLevel = false;

tp.Text = \_frm.Text;

tabControl2.TabPages.Add(tp);

tp.Controls.Add(\_frm);

\_frm.Show();

tabControl2.SelectedTab = tp;

}

}

Ez annyit tesz, hogy egy függvény visszatérési értékével eldönti, hogy a Form aktív-e már vagy sem. Ha nem aktív akkor nyissa meg a Formot és Dockolja be a tabControl2-re, azon belül pedig az éppen létrehozott TabPage-re.

private bool formActive(Form \_frm) {

if (tabControl2.TabCount > 0) {

foreach (TabPage item in tabControl2.TabPages) {

if (\_frm.Text == item.Text) {

tabControl2.SelectedTab = item;

return true;

}

}

}

return false;

}

Ha a formActive függvénynek egy olyan Formot adunk át amelyiknek a neve már szerepel a tabControl2.TabPagek között, akkor kiválasztja az adott tabPaget. A megnyitott dockolt formokat a Main Formon létrehozott statikus metódussal tudjuk bezárni. Ami vár egy string paramétert, ez lesz a megnyitott Form neve. (Remélhetőleg nem lesz 2 ugyan olyan oldalnév a jövőben!)

public static void closeTabpage(string szoveg) {

foreach (TabPage tp in staticControl2.TabPages) {

if (tp.Text == szoveg) {

staticControl2.TabPages.Remove(tp);

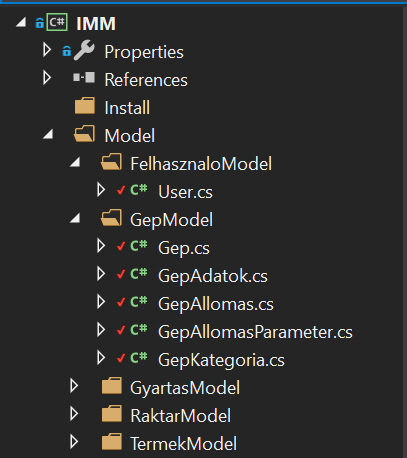
}

}

}

## Osztályok és metódusok

A programban annyi osztály szerepel a MODELL mappában ahány adatbázis tábla van. Minden táblának saját MODELL-je van a forrásban.

  
*(3. Kép. Solution Explorer)*

Minden egyes osztálynak van egy getAll() metódusa. Ennek a metódusnak az a szerepe, hogy az adatbázisból kiolvassa az adatokat. Ez egy publikus statikus List<OSZTÁLY> getAll() lista. Így a visszatérési listát példányosítás nélkül is használni lehet.

public static List<Termek> getAll() {

List<Termek> \_termekek = new List<Termek>();

SQLiteCommand sqlcommand = new SQLiteCommand(sqlc);

SQLiteDataReader dr;

try {

sqlc.Open();

sqlcommand.CommandText = "SELECT \* FROM Termekek";

dr = sqlcommand.ExecuteReader();

while (dr.Read()) {

Termek jelenlegiTermek = new Termek(Convert.ToInt32(dr.GetValue(0)), dr.GetValue(1).ToString(), dr.GetValue(2).ToString(), dr.GetValue(3).ToString(), Convert.ToInt32(dr.GetValue(4)), Convert.ToInt32(dr.GetValue(5)), Convert.ToInt32(dr.GetValue(6)), Convert.ToInt32(dr.GetValue(7)), Convert.ToInt32(dr.GetValue(8)));

\_termekek.Add(jelenlegiTermek);

}

dr.Close();

} catch (Exception ex) {

MessageBox.Show(ex.Message, "Termékek kiolvasása SQL hiba!");

Logger.Log("Termek getAll", ex.Message);

}

Database.checkConnectionState(sqlc);

return \_termekek;

}

Mivel innentől kezdve egy listával tudunk dolgozni, így lambda és linq-val tudunk lekérdezéseket készíteni. A legtöbb osztálynak vannak beágyazott lekérdezése, amiket szintén példányosítás nélkül tudunk használni. Ilyen például a Termek.findByID(int id). Ez az alábbi módon néz ki: public static Termek findByID(int id) {

Termek \_termek = (from x in getAll()

where x.id == id

select x).FirstOrDefault();

return \_termek;

}

Itt a visszatérési érték már nem lista lesz, mivel ugye egy adott ID csak egyszer szerepelhet a keresett adatbázisban. Ezeken kívül egyes osztályoknak, ahol lehetőség van szerkesztésre/hozzáadásra/eltávolításra, ott ezeket metódusokkal tudjuk megoldani, példányosítás nélkül. Ez az alábbi módon néz ki: public static void Modosit(Termek \_termek) {

SQLiteCommand sqlcommand = new SQLiteCommand(sqlc);

sqlc.Open();

try {

sqlcommand.CommandText = "UPDATE Termekek SET termekNev='" + \_termek.TermekNev + "',termekNev2='" + \_termek.TermekNev2 + "',cikkszam='" + \_termek.Cikkszam + "',kategoria='" + \_termek.KategoriaID + "',csomagolasidarabszam='" + \_termek.CsomagolasiDarabszam + "',minimumgyarthato='" + \_termek.MinimumGyarthato + "',aktiv='" + \_termek.Aktiv + "',raktar='" + \_termek.Raktar + "' where id='" + \_termek.ID + "'";

sqlcommand.ExecuteNonQuery();

} catch (Exception ex) {

MessageBox.Show(ex.Message, "Termek módosítása SQL hiba!");

Logger.Log("Database", ex.Message);

}

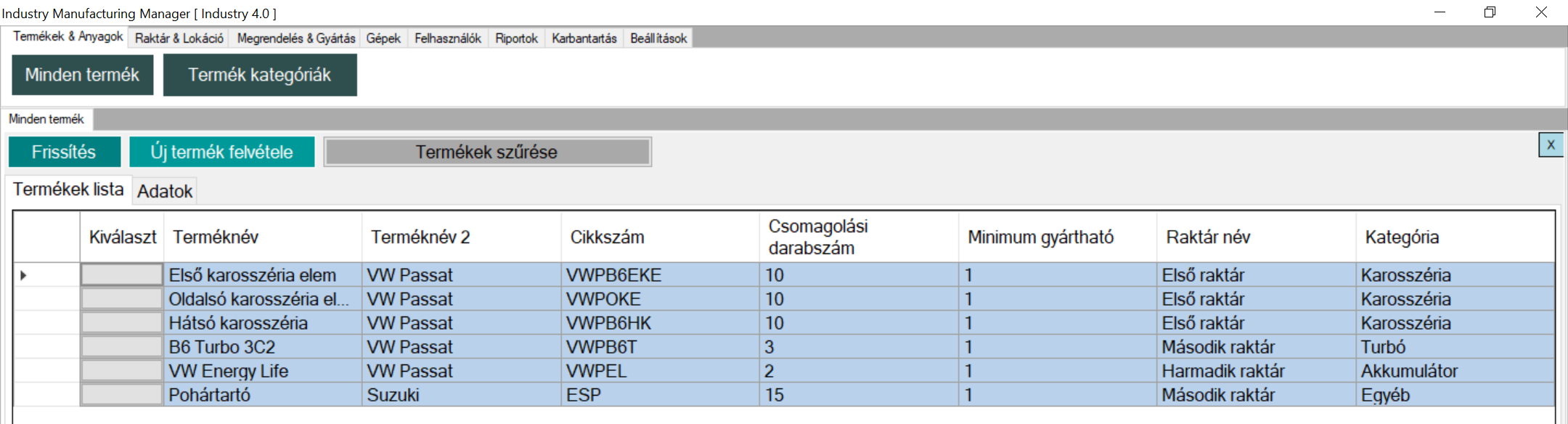
Database.checkConnectionState(sqlc);

}

## Megjelenítőeszközök ( GridView, TreeView, ComboBox)

### GridView

Sok helyen találhatóak különböző megjelenítőeszközök, például GridView-ek, treeView-ek. A termék modulban megjelenített Termékek GridView-en lehetőségünk van szűrni is a megjelenített adatokra, egy „Szűrés” gombra kattintás után. Az felsorolt megjelenítőeszközök mind írásvédettek. A megjelenített adatokat csak úgy lehet módosítani, hogy a módosítani kívánt adat „Kiválaszt” oszlopában lévő gombra rányomunk. Ennek hatására megnyílik a kiválasztott adat lapja. Itt már lehet módosítani, hozzáadni, eltávolítani.

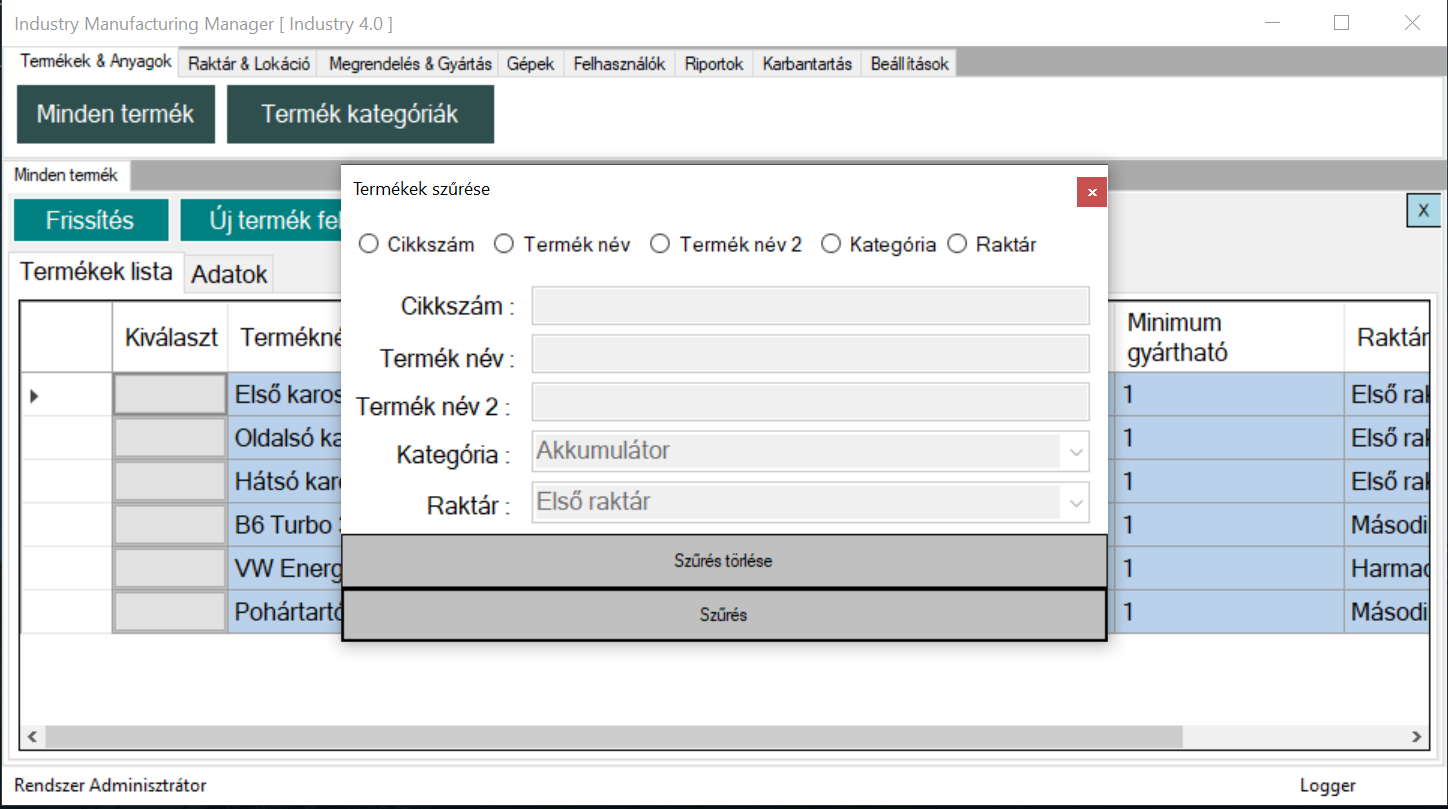
  
*(4. Kép Termékek GridView)*

Ha egy GridView-en minden adatot meg szeretnénk jeleníteni ( mint itt a Minden Termék GridView-en) ,akkor ezt a forrásban az osztályon belül létrehozott getAll() statikus listával tehetjük meg. Jelenleg ez így működik a Minden Termék esetében :

A Form\_Load eseménynél feltöltöm a List<Termek> termekekLista litát.

termekekLista = Termek.getAll();

Majd egy gridFeltolt() metódussal elrejtem a nem szükséges Oszlopokat, illetve a termekekGrid.DataSource = termekekLista; lesz. Így az oldal betöltésekor egyből látjuk az összes terméket. Ez után átnevezi az oszlopneveket, hogy ne a Modellben leírt változók nevét használják az oszlopok. Ha szűrni szeretnénk bizonyos adatokra, amit itt a Minden Termékek oldalon meg is tudunk tenni, akkor szintén a modell.getAll() metódust fogja alapul venni, viszont Lambda és Linq-val szűrni fogjuk a listát. A szűrő oldal így néz ki :

  
*(5. Kép Szűrő Form)*

Amint egy RadioButton-t bejelölünk írható lesz az ahhoz tartozó TextBox. Majd ha beírtunk egy értéket a mezőbe és rányomunk a „Szűrés” gombra, azonnal változik a háttérben lévő lista. Ennek a szűrési Formnak a konstruktorában meg van adva egy DataGridView Ennek átadjuk az összes Termékek oldalon található DataGridView-et, majd amint rányomunk a szűrés gombra, a modellből getAll() metódussal kinyerjük az összes terméket és szűrünk benne. Például

if (cikkCheck.Checked) {

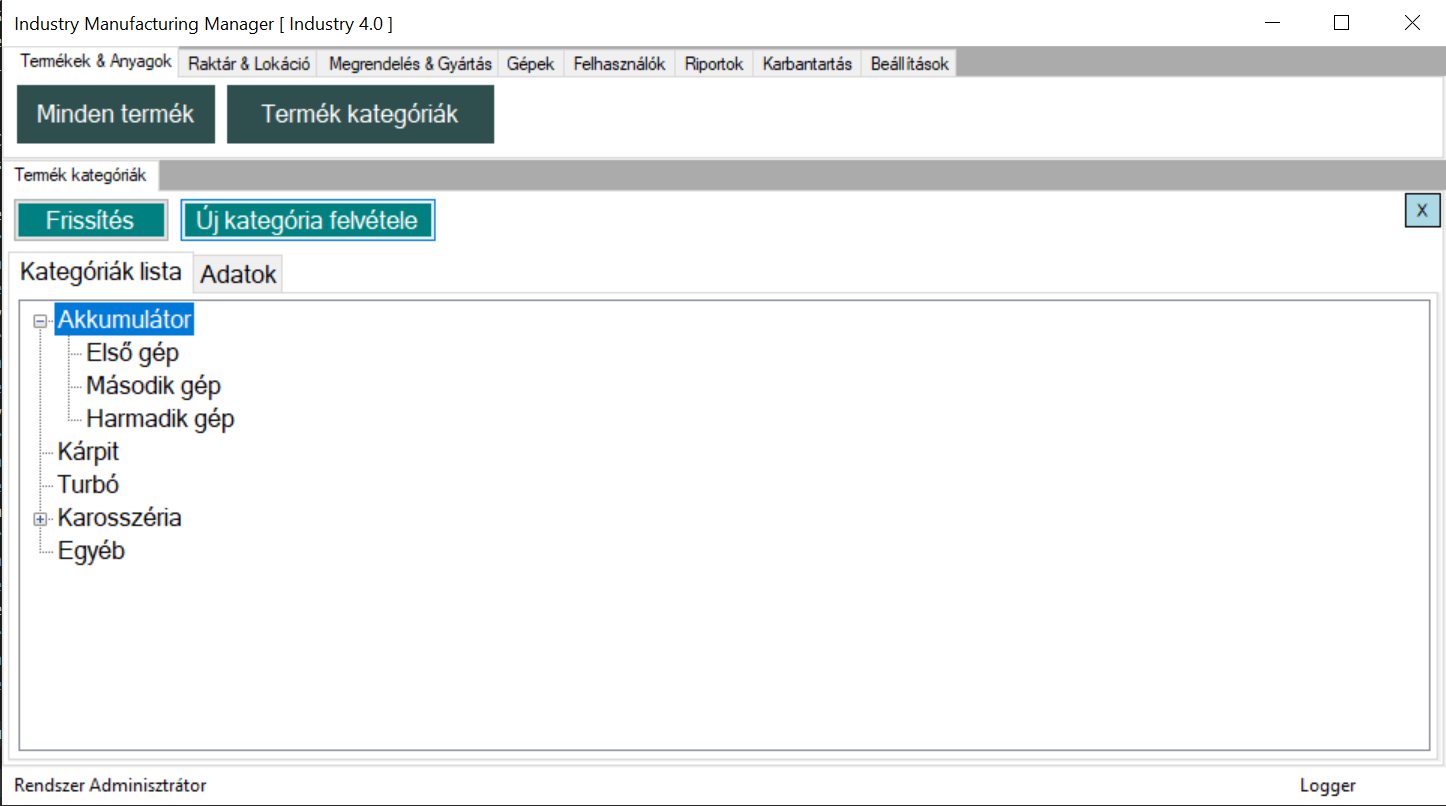
szurtLista = (from x in Model.Termek.getAll()

where x.Cikkszam.Contains(cikkszamTextbox.Text)

select x).ToList();   
dgv.DataSource = szurtLista;

Így a háttérben lévő termékek Gridje azonnal változik a szűrt adatoktól függően.

### TreeView Találhatóak TreeView-ek a programban. Például a Termék Kategóriák menüpont alatt. Az említett TreeView így néz ki:

  
*(6. Kép Kategória TreeView)*

Az alábbi nézet feltöltése úgy működik, hogy amikor egy gépet viszünk fel a rendszerbe ott szükséges kiválasztani egy Kategóriát. A kiválasztott kategória ID-je hozzá lesz társítva az adott géphez. Így amikor új gépet viszek fel és megadok rá egy kategóriát, akkor az a kategória lesz a „Szülő”. A nézet feltöltése így néz ki :

void populateTreeview() {

kategoriakTree.Nodes.Clear();

List<Kategoria> katList = Kategoria.getAll();

try {

foreach (var kategoria in katList) {

List<GepKategoria> szurtGepkategoriak = (from x in gepKategoria.getAll()

where x.KategoriaId == kategoria.Id

orderby x.GepId

select x).ToList();

kategoriakTree.Nodes.Add(kategoria.KategoriaNev);

if (szurtGepkategoriak.Count != 0) {

foreach (var szgepkategoria in szurtGepkategoriak) {

kategoriakTree.Nodes[kategoria.Id-1].Nodes.Add(szgepkategoria.GepNev);

}

} else {

}

}

} catch (Exception ex) {

MessageBox.Show(ex.Message, "populateTreeError");

}

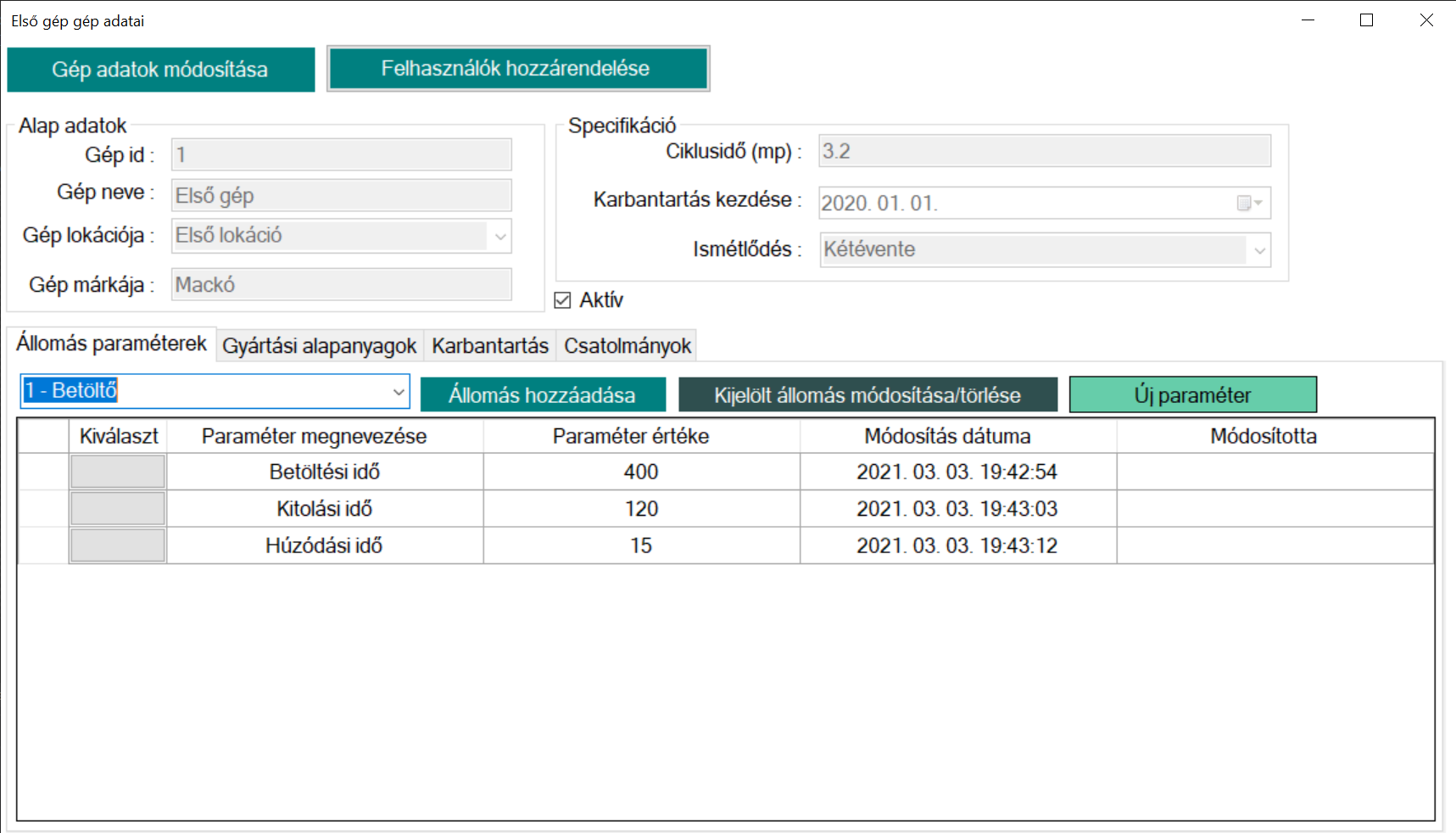
}

A metódus elején elsőnek töröljük a korábban feltöltött nézetet. Utána létrehoz egy listát amiben minden Kategória szerepel. Egy foreach ciklussal kikeressük a kategóriákat a megadott listából és minden kategóriára létrehozunk egy új szűrt gép kategóriák listát. Majd a TreeView-hez fogjuk adni elsőnek a kategória nevét, majd ha a kategória alá tartoznak gépek, akkor azokat a gépeket is, de a szülő az előzőleg hozzáadott Node lesz.

kategoriakTree.Nodes[kategoria.Id-1].Nodes.Add(szgepkategoria.GepNev);

### ComboBox

A gépek megjelenítő felületén található ComboBox és GridView összekapcsolás. Ez csak annyit jelent, hogy amikor egy másik értéket választunk ki a ComboBoxban, akkor a GridView ismét változik. Viszont, maga a ComboBox tartalma is a kiválasztott géptől függ, ugyanis ezek a ComboBoxot tartalmazzák a gépnek az állomás megnevezéseit, az ehhez kapcsolódó GridView pedig az állomások alá tartozó paramétereket.

  
*(7.Kép Gép ComboBox és GridView kapcsolat)*

Az állomás ComboBox ( a 7.Kép Gép ComboBox és GridView kapcsolat képen 1 – Betöltő van kiválasztva) feltöltése a GepAllomas Modell szerint történik. Viszont mindig az aktuális gépre felvitt állomásokat láthatjuk itt. Ezt LinQ-val keresi ki a program a List<GepAllomas> listában. Azok az állomások lesznek visszaadva ahol a GepId = gepidTextbox.Text-tel. Az ez alatt található GridView pedig a kiválasztott állomás paraméterei. Tehát ha mondjuk idő közben a gép kopó alkatrészei miatt paramétereket kell változtatni a kiválasztott gépen, akkor itt nyomon lehet követni mikor történt változás és milyen paraméterre történt a változtatás. Így a gépkezelők egy esetleges rossz beállítás után könnyen kikereshetik az előző, még működő paramétert. Ez a GridView pedig a kiválasztott állomás ID-je szerint fogja feltölteni a GridView-et. Ez a feltöltési metódus minden SelectedIndexChanged eseménynél lefut. Így amikor egy új állomást választunk ki, akkor már töltődik is a Grid.

allomasParameterGrid.DataSource=GepAllomasParameter.findByAllomasID(Convert.ToInt32(allomasokComboBox.SelectedValue));

## Frissítések

Jelenleg a szoftver nem kezeli a frissítéseket. GitHub verziókövető rendszerrel publikáltam a megtörtént frissítéseket. Jövőbeli elképzelés az lenne, hogy egy weboldalon közzéteszek egy szöveges dokumentumot. Ebben nem lesz semmi egyéb info csak a verzió száma. A program minden indításkor egy WebRequest osztály segítségével ellenőrzi a verziószámot, ha a weboldalon lévő nagyobb, akkor frissíti a programot, illetve kicseréli a jelenlegi verziószámot.

## Tovább fejlesztés és ötletek

* A jelenleg nem használt oszlopok használata. Ilyen oszlopok például az addedby, active. Jelenleg a törlés gombok valóbal törlik az adatbázisból az adatokat, ezt szeretném elkerülni a jövőben és csak „inaktívvá” tenni, így könnyebben vissza lehet állítani az esetleges felhasználói hibából történt törléseket.
* Csatolmányok hozzáadása minden modulhoz.(Termék,Raktár,Felhasználó,Gép)
* Felhasználói jogosultságok, belépési szintekkel.
* Karbantartási előrejelzés.