# Bevezető

Leltárrendszer, adatbázis kezelő program.

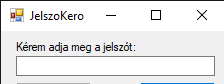
Készítette: Balogh József

A cégeknek nagyon fontos az, hogy mindenről tudjanak rekordokat használni és ezt általában egy adatbázis segítségével szokták megoldani, viszont nem mindenki tud adatbázist kezelni. Ez miatt egy leltárrendszer program egy cégen belűl nagyon fontos tud lenni adatok kezelésében, módosításában.

# Felhasználói Dokumentáció

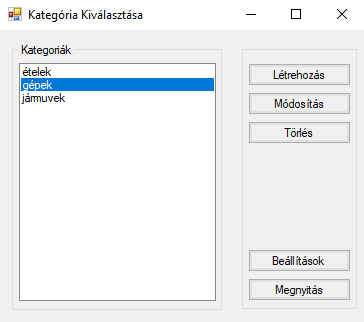
## Jelszó védelem

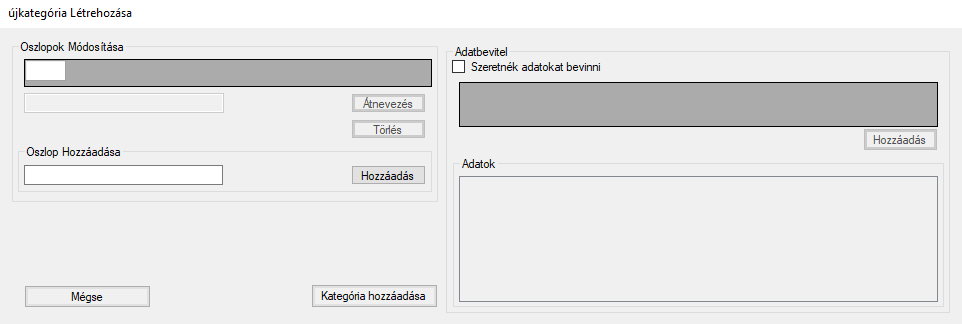
A program minden mentéskor és indításkor a jelszót kérni fogja. Ez azért kell, hogy olyan felhasználó ne tudjon változtatni az adatok közül, aki nincs arra felhatalmazva.

Legelső indításkor ez „root”, de beállításokban meg lehet változtatni.

## Kategóriák

Egy cégnek nagyon fontos ha az adatai kategorizálva vannak. Ezáltal könnyebb áttekinteni és megkeresni az adatokat.

Külömböző kategóriákat ki tudja választania a felhasználó, ezáltal hozzá tud férni az adott kategóriában lévő adatokhoz illetve módosítani tudja azokat. A program elindításkor fel fogja hozni az elérhető kategóriákat amelyhez hozzá tud férni, vagy új kategóriát tud létrehozni.

A felhasználó tud új kategóriákat létrehozni. Létrehozáskor a program kérni fogja a kategória nevét és az oszlopok nevét. Ha a felhasználó kiválasztotta azt az opciót hogy a program vigyen be adatokat az új kategóriába úgy hogy ne vigye át a létrehozott kategóriába akkor a program kérni fogja azokat az adatokat.

A felhasználó tudja módosítani a meglévő kategóriákat kategóriaválasztás ablakban a „Módosítás” gombra kattintva. Szerkeszteni tudja a kategória nevét, oszlopok nevét, illetve tud hozzáadni vagy eltávolítani oszlopokat.

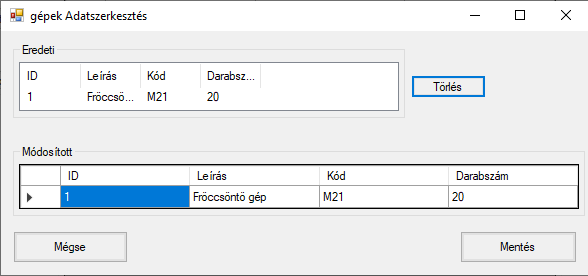
## Adatkezelés

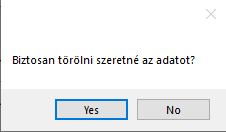
## 

Miután a felhasználó kiválasztja a kategóriát egyből a kategóriában lévő adatokat fogja látni ahol hozzá tud adni, törölni, szerkeszteni vagy keresni tud az adatok között.

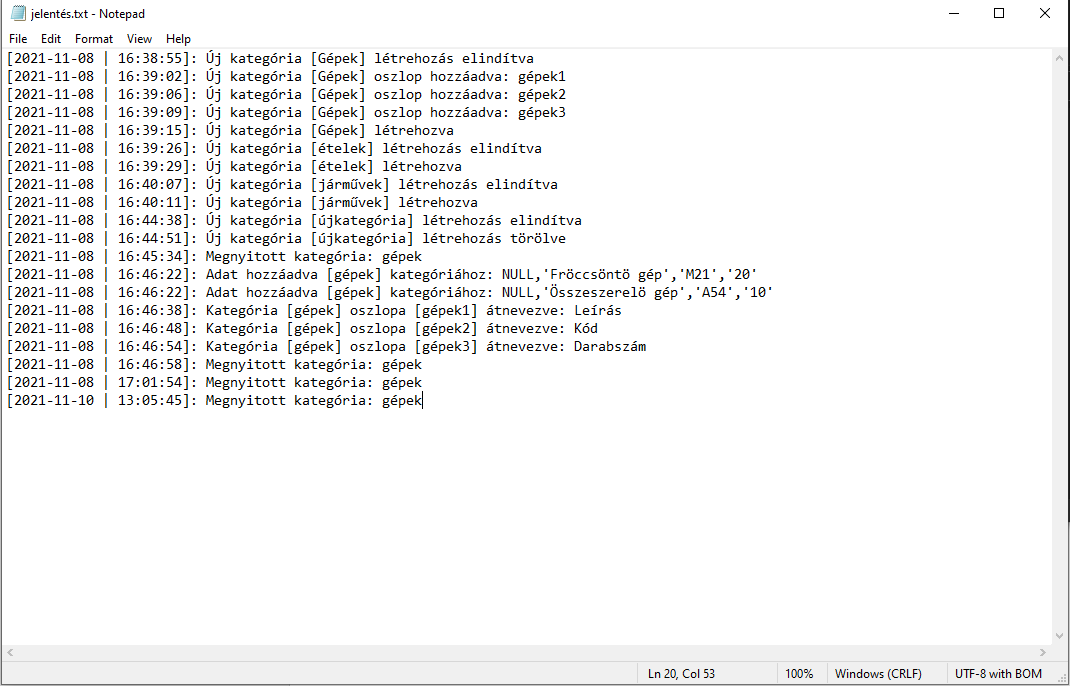
Hozzáadásnál a program be fogja kérni azokat az adatokat amelyek kellenek hogy be tudja vinni az adatbázisba. Ezek lehetnek leírások, darabszámok. A „Hozzáadás” gombra kattintva az adatokat a program beviszi az adatbázisba.

A felhasználó az adatra kétszer rákattintva megkapja a lehetőséget hogy törölni vagy szerkeszteni tudja.



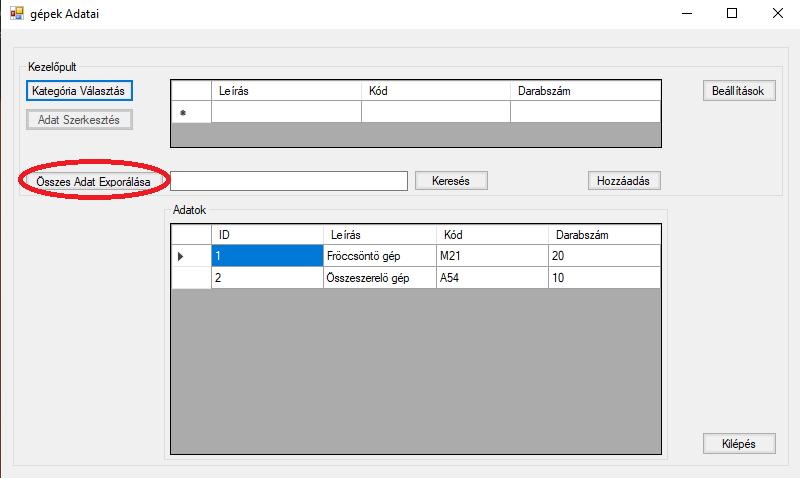
Törlés esetén a program még egyszer rá fog kérdezni hogy biztosan akarja törölni véletlen törlések elkerülése érdekében. Szerkesztéskor a mentés gombra kattintással fogja elmenteni a megváltoztatott adatot. Mégse gombbal lehet visszalépni.

## Jelentés

A program minden módosításkor egy jelentés nevű txt fájlba ki fogja írni ami a program mappájában tallható, hogy mik, mivé és mikor lettek megváltoztatva.

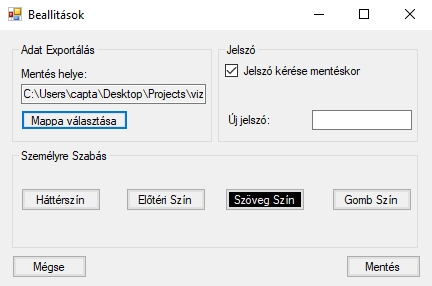
## Adatexportálás

A felhasználó ki tudja exportálni a kiválasztott kategóriában lévő adatokat egy csv formátumú fájlba az Adatok ablakon található „Összes Adat Exportálása” gombra nyomva.



A programnak Beállításokon belűl lehet megadni, hogy hova exportálja az adatokat

## Beállítások

A felhasználó beállításokon belűl tudja személyre szabni a programot, a jelszót megváltoztatni illetve az adat exportálásnak az útvonalát megváltoztatni.

# Fejlesztői Dokumentáció

## Bevezető

Szükséges olyan operációs rendszer ami .exe fájlokat tud futtatni, grafikus felületű és a .NET keretrendszer telepítve van. Illetve C# fordítóprogram szükséges.

## Adatbázis

A program folyamatosan kapcsolatot tart a „leltarrendszer” nevű adatbázissal működés közben. Az adatbázis feltelepítéséhez a következő SQL parancsok szükségesek:

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

START TRANSACTION;

SET time\_zone = "+00:00";

CREATE DATABASE IF NOT EXISTS `leltarrendszer` DEFAULT CHARACTER SET utf8 COLLATE utf8\_hungarian\_ci;

USE `leltarrendszer`;

COMMIT;

Illetve az adatbázis eléréséhez a program egy felhasználói fiókot használ. Feltelepítéséhez szükséges SQL parancsok a következők:

CREATE USER 'leltarrendszer'@'localhost' IDENTIFIED BY 'Jelszo123!';

GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, SHOW VIEW ON \*.\* TO 'leltarrendszer'@'localhost';

## Kategóriák

A program indításkor ez a legelső ablak ami feljön jelszóbekérés után. Legelső indításkor a jelszó „root”.

A kategóriák a program fő adatbázisában lévő elérhető táblanevek kiíratása a ListBox-ban. Illetve adatbázis csatlakozás ellenőrzése, ha nem sikerült kapcsolatot létesíteni a program kilép.

public void loadCategory()

{

listBox1.Items.Clear();

try

{

Program.conn.Open();

string sqlCommand = "select table\_name from information\_schema.tables where table\_schema = 'leltarrendszer'";

MySqlCommand cmd = new MySqlCommand(sqlCommand, Program.conn);

cmd.CommandType = CommandType.Text;

MySqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

listBox1.Items.Add(rdr["table\_name"]);

}

rdr.Close();

}

catch

{

Program.writeReport($"SQL Adatbázis csatlakozási hiba");

MessageBox.Show("Nem lehet csatlakozni az adatbázishoz!\nAz alkalmazás kilép.");

Application.Exit();

}

Program.conn.Close();

}

A megnyitás egy parancsot fog visszaküldeni a fő Formnak, hogy miket kell betöltenie az adatbázisból.

//Megnyitás

private void button3\_Click(object sender, EventArgs e)

{

Program.category = listBox1.Items[listBox1.SelectedIndex].ToString();

Program.writeReport($"Megnyitott kategória: {Program.category}");

this.Close();

}

## Kategóriák Létrehozása

Létrehozásnál egy új táblát fog elkészíteni a program a megadott oszlopokkal az SQL adatbázisban, valós időben, illetve több ellenőrzés is megtörténik, hogy ne lehessen olyan oszlopnevet adni amit az adatbázis nem fogad el.

//Hozzáadás

private void add()

{

string colName = textBox2.Text;

if (char.IsDigit(colName[0]))

{

MessageBox.Show("Az oszlop neve nem kezdőthet számmal!");

return;

}

if (colName.Contains(" "))

{

MessageBox.Show("Az oszlop neve nem tartalmazhat szóközt!");

}

else

{

colName = Regex.Replace(colName, @"[^0-9a-zA-ZÖöÜüÓóŐőÚúÉéÁáŰűÍí]+", "");

Program.conn.Open();

string sqlCommand = "SHOW COLUMNS FROM " + Program.newCategory;

MySqlCommand cmd = new MySqlCommand(sqlCommand, Program.conn);

cmd.CommandType = CommandType.Text;

MySqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

if (rdr["Field"].ToString().Contains(colName))

{

MessageBox.Show("Ilyen oszlop már létezik!");

rdr.Close();

Program.conn.Close();

return;

}

}

rdr.Close();

Program.conn.Close();

Program.conn.Open();

string sqlCommand2 = "ALTER TABLE " + Program.newCategory + " ADD COLUMN " + colName + " TEXT AFTER " + Columns.Last();

MySqlCommand cmd2 = new MySqlCommand(sqlCommand2, Program.conn);

cmd.CommandType = CommandType.Text;

MySqlDataReader rdr2 = cmd2.ExecuteReader();

rdr2.Close();

Program.conn.Close();

Program.writeReport($"Új kategória [{Program.newCategory}] oszlop hozzáadva: {colName}");

loadCategoryCols();

loadData();

}

}

Illetve, ha adatot akar feltölteni a felhasználó, úgy, hogy ne vigye bele egyből abba a kategóriába, akkor a megadott oszlopokkal fogja bekérni adatokat a „Szeretnék adatokat bevinni” checkbox kipipálása után.

private List<string> ColumnsData = new List<string>();

DataTable dtData = new DataTable();

static List<Sor> sorok = new List<Sor>() { };

private void loadData()

{

if (Program.newCategory != null)

{

dtData.Rows.Clear();

dtData.Columns.Clear();

sorok.Clear();

listView1.Clear();

ColumnsData.Clear();

Program.conn.Open();

string sqlCommand = "SHOW COLUMNS FROM " + Program.newCategory;

MySqlCommand cmd = new MySqlCommand(sqlCommand, Program.conn);

cmd.CommandType = CommandType.Text;

MySqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

ColumnsData.Add(rdr["Field"].ToString());

}

rdr.Close();

foreach (string col in Columns)

{

listView1.Columns.Add(col);

dtData.Columns.Add(col);

}

MySqlCommand selectAllRow = new MySqlCommand("SELECT \* FROM " + Program.newCategory, Program.conn);

selectAllRow.CommandType = CommandType.Text;

MySqlDataReader rdr2 = selectAllRow.ExecuteReader();

while (rdr2.Read())

{

Sor s = new Sor();

for (int i = 0; i < rdr2.FieldCount; i++)

{

s.AddData(rdr2.GetValue(i).ToString());

}

sorok.Add(s);

}

rdr2.Close();

Program.conn.Close();

for (int i = 0; i < sorok.Count; i++)

{

listView1.Items.Add(new ListViewItem(sorok[i].data.ToArray<string>()));

}

dataGridView2.DataSource = dtData;

dataGridView2.Columns[0].Visible = false;

foreach (DataGridViewColumn col in dataGridView2.Columns)

{

col.AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

}

}

}

## Kategóriák Törlése

Kategória törlése a kategória választó ablakból történik ahol a felhasználót újból megkérdezi a program, hogy biztos-e a törlésben végrehajtás előtt. Majd sikeres törlés után a felhasználót arról értesíti.

//Törlés

private void button4\_Click(object sender, EventArgs e)

{

DialogResult dialogResult = MessageBox.Show("Biztosan törölni szeretné az adatot?", "", MessageBoxButtons.YesNo);

if (dialogResult == DialogResult.Yes)

{

if (Program.askPassword)

{

JelszoKero askPAss = new JelszoKero();

askPAss.ShowDialog();

if (Program.passwordCorrect == false)

{

return;

}

}

Program.conn.Open();

string sqlCommand = "DROP TABLE " + listBox1.Items[listBox1.SelectedIndex].ToString();

MySqlCommand cmd = new MySqlCommand(sqlCommand, Program.conn);

cmd.CommandType = CommandType.Text;

MySqlDataReader rdr = cmd.ExecuteReader();

rdr.Close();

Program.conn.Close();

Program.writeReport($"Kategória törölve: {listBox1.Items[listBox1.SelectedIndex].ToString()}");

MessageBox.Show("Kategória törölve.");

button4.Enabled = false;

button1.Enabled = false;

button1.Enabled = false;

loadCategory();

}

}

## Kategóriák Módosítása

Kategória szerkesztés ablakban lehet átnevezni, törölni vagy hozzáadni oszlopokat.

Átnevezéskor a program ellenőrzi a bevitt oszlop nevét SQL parancs kiadása előtt, hogy nem-e tartalmaz olyan karaktert amit az SQL adatbázis nem tud kezelni.

//Átnevezés

private void rename()

{

string oldname = dataGridView1.Columns[colIndex].HeaderText.ToString();

string newname = textBox1.Text;

newname = Regex.Replace(newname, @"[^0-9a-zA-ZÖöÜüÓóŐőÚúÉéÁáŰűÍí]+", "");

if (oldname == newname)

{

MessageBox.Show("Új név nem lehet ugyan az mint a régi név.");

}

else

{

Program.conn.Open();

string sqlCommand = "ALTER TABLE " + categoryToChange + " CHANGE COLUMN " + oldname + " " + newname + " TEXT";

MySqlCommand cmd = new MySqlCommand(sqlCommand, Program.conn);

cmd.CommandType = CommandType.Text;

MySqlDataReader rdr = cmd.ExecuteReader();

rdr.Close();

Program.conn.Close();

Program.writeReport($"Kategória [{Program.category}] oszlopa [{oldname}] átnevezve: {newname}");

loadCategoryCols();

}

}

Törlésnél a program ismételten meg fogja kérdezni a felhasználót, hogy biztos-e a törlésben végrehajtás előtt. Majd sikeres törlés után a felhasználót arról értesíti.

//Törlés

private void button2\_Click(object sender, EventArgs e)

{

DialogResult dialogResult = MessageBox.Show("Biztosan törölni szeretné az adatot?", "", MessageBoxButtons.YesNo);

if (dialogResult == DialogResult.Yes)

{

Program.conn.Open();

string sqlCommand = "ALTER TABLE " + categoryToChange + " DROP COLUMN " + dataGridView1.Columns[colIndex].HeaderText.ToString();

MySqlCommand cmd = new MySqlCommand(sqlCommand, Program.conn);

cmd.CommandType = CommandType.Text;

MySqlDataReader rdr = cmd.ExecuteReader();

rdr.Close();

Program.conn.Close();

Program.writeReport($"Kategória [{Program.category}] oszlopa [{dataGridView1.Columns[colIndex].HeaderText.ToString()}] törölve");

loadCategoryCols();

}

}

Oszlop hozzáadásnál a program kódja ugyan az mint kategória létrehozásnál, itt is leelenőrzi a bevitt oszlop nevét SQL parancs kiadása előtt.

Illetve oszlop törlés, átnevezés új kategória létrehozásnál is jelen van, hogy a felhasználó helyben tudja módosítani az új kategóriát.

## Adatok

Az adatok egy ListView-ben vannak kiírva a kategóriaválasztás ablakból kiválaszott kategória (table)-ből, az összes adat egy objektumként szerepel.

private void loadData()

{

if (Program.category != null)

{

this.Text = $"{Program.category} Adatai";

dt.Rows.Clear();

dt.Columns.Clear();

sorok.Clear();

dtDisplay.Rows.Clear();

dtDisplay.Columns.Clear();

Columns.Clear();

dtDisplay.DefaultView.Sort = string.Empty;

Program.conn.Open();

string sqlCommand = "SHOW COLUMNS FROM " + Program.category;

MySqlCommand cmd = new MySqlCommand(sqlCommand, Program.conn);

cmd.CommandType = CommandType.Text;

MySqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

Columns.Add(rdr["Field"].ToString());

}

rdr.Close();

foreach (string col in Columns)

{

dtDisplay.Columns.Add(col);

dt.Columns.Add(col);

}

MySqlCommand selectAllRow = new MySqlCommand("SELECT \* FROM " + Program.category, Program.conn);

selectAllRow.CommandType = CommandType.Text;

MySqlDataReader rdr2 = selectAllRow.ExecuteReader();

while (rdr2.Read())

{

Sor s = new Sor();

for (int i = 0; i < rdr2.FieldCount; i++)

{

s.AddData(rdr2.GetValue(i).ToString());

}

sorok.Add(s);

}

rdr2.Close();

Program.conn.Close();

for (int i = 0; i < sorok.Count; i++)

{

dtDisplay.Rows.Add(sorok[i].data.ToArray<string>());

}

dataGridView2.DataSource = dtDisplay;

dataGridView1.DataSource = dt;

dataGridView1.Columns[0].Visible = false;

foreach (DataGridViewColumn col in dataGridView2.Columns)

{

col.AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

}

foreach (DataGridViewColumn col in dataGridView1.Columns)

{

col.AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

}

}

}

## Adatok Hozzáadása

Hozzáadásnál elsőnek csak kijelenésnél változik az adatlista, ahol egy új objektumot hoz csak létre, a mentésnél fogja az összes új rekordot feltölteni az adatbázisba. Feltöltés előtt a program a feltöltendő adatokból kitörli az összes speciális karaktert pl: - | \ /. Illetve ellenőrzi, hogy a felhasználó adott-e új adatot amit fel szeretne tölteni.

//Hozzáadás

private void button4\_Click(object sender, EventArgs e)

{

if (!(dt.Rows.Count > 0))

{

MessageBox.Show("Kérem adjon meg adatot.");

return;

}

for (int i = 0; i < dt.Rows.Count; i++)

{

Sor newRow = new Sor();

for (int j = 0; j < dt.Rows[i].ItemArray.Count(); j++)

{

newRow.AddData("'" + Regex.Replace(dt.Rows[i].ItemArray[j].ToString(), @"[^0-9a-zA-ZÖöÜüÓóŐőÚúÉéÁáŰűÍí ]+", "").Trim() + "'");

}

newRow.data[0] = null;

string allCols = string.Join(",", Columns.Select(x => x.ToString()).ToArray());

string rowData = string.Join(",", newRow.data.ToArray());

Program.conn.Open();

string sqlCommand = "INSERT INTO " + Program.category + " (" + allCols + ") VALUES (NULL " + rowData + ")";

MySqlCommand cmd = new MySqlCommand(sqlCommand, Program.conn);

cmd.CommandType = CommandType.Text;

MySqlDataReader rdr = cmd.ExecuteReader();

rdr.Close();

Program.conn.Close();

newRow.data[0] = "NULL";

string rowDataRep = string.Join(",", newRow.data.ToArray());

Program.writeReport($"Adat hozzáadva [{Program.category}] kategóriához: {rowDataRep}");

}

loadData();

MessageBox.Show("Adat hozzáadva.");

}

## Adatok Keresése

A keresés funkció kikeresi az adatlistából azokat az adatokat, ahol egyezik a keresett szöveggel. Ha rákeresünk egy adatra, akkor visszakapunk egy új Listát, ahol csak azok a rekordok vannak, ahol a keresett szöveg, illetve szövegrészlet található.

private void search()

{

string keresndo = textBox1.Text.ToLower();

List<Sor> query = new List<Sor>();

for (int i = 0; i < sorok.Count; i++)

{

if (sorok[i].data.ConvertAll(x => x.ToLower()).Contains(keresndo) || sorok[i].data.ConvertAll(x => x.ToLower()).Any(x => x.Contains(keresndo)))

{

query.Add(sorok[i]);

}

}

dtDisplay.Rows.Clear();

if (keresndo.Trim() == "")

{

for (int i = 0; i < sorok.Count; i++)

{

dtDisplay.Rows.Add(sorok[i].data.ToArray<string>());

}

}

else

{

for (int i = 0; i < query.Count; i++)

{

dtDisplay.Rows.Add(query[i].data.ToArray<string>());

}

}

query.Clear();

}

## Adatexportálás

Az adatexportálás az „Összes Adat Exportálása” gombra nyomva a kiválasztott kategóriát kimenti egy CSV fájlba, amit egy táblaszerkesztő program tudja olvasni pl: Excel. Illetve a program exportláls után megkérdezi a felhasználót, hogy az exportált CSV fájl mappáját meg-e nyissa. Megerősítés után a program megnyissa a fájlkezelőt a megfelelő elérési úttal.

//Adatexportálás

private void button5\_Click(object sender, EventArgs e)

{

string csv = string.Empty;

foreach (DataGridViewColumn column in dataGridView2.Columns)

{

csv += column.HeaderText + ",";

}

csv += "\r\n";

foreach (DataGridViewRow row in dataGridView2.Rows)

{

foreach (DataGridViewCell cell in row.Cells)

{

csv += cell.Value.ToString().Replace(",", ";") + ",";

}

csv += "\r\n";

}

File.WriteAllText($"{Program.DataExportPath}/{Program.category} {DateTime.Now.ToString("yyyy.MM.dd")}.csv", csv, Encoding.UTF8);

DialogResult dialogResult = MessageBox.Show("Adatok exportálva.\nMeg szeretné nyitni a tartalmazó mappát?", "", MessageBoxButtons.YesNo);

if (dialogResult == DialogResult.Yes)

{

System.Diagnostics.Process.Start("explorer.exe", Program.DataExportPath);

}

}

## Adatok Módosítása, Törlése

Kétszer rákattintva egy adatra, vagy az adat kiválasztása után „Adat Szerkesztése” gombbra nyomva a program felhoz egy új ablakot, ahol a felhasználó adatot tudja módosítani, ez lehet az adatnak egyik értéke, vagy a törlése. A „Mentés” gombra kattintva, a program egy ellenőrzés után felviszi az adatbázisba.

//Mentés

private void button1\_Click(object sender, EventArgs e)

{

if (Program.askPassword)

{

JelszoKero askPAss = new JelszoKero();

askPAss.ShowDialog();

if (Program.passwordCorrect == false)

{

return;

}

}

List<string> stringList = new List<string>();

for (int i = 0; i < Columns.Count; i++)

{

stringList.Add(Columns[i] + "='" + Regex.Replace(dt.Rows[0].ItemArray[i].ToString(), @"[^0-9a-zA-ZÖöÜüÓóŐőÚúÉéÁáŰűÍí ]+", "").Trim() + "'");

}

string allEditedItems = string.Join(",", stringList.Select(x => x.ToString()).ToArray());

Program.conn.Open();

string sqlCommand = "UPDATE " + Program.category + " SET " + allEditedItems + " WHERE ID =" + dt.Rows[0].ItemArray[0];

MySqlCommand cmd = new MySqlCommand(sqlCommand, Program.conn);

cmd.CommandType = CommandType.Text;

MySqlDataReader rdr = cmd.ExecuteReader();

rdr.Close();

Program.conn.Close();

Program.writeReport($"Adat [{string.Join(",", kivalasztottSor.data)}] megváltoztatva [{Program.category}] kategóriában: {string.Join(",", stringList)}");

MessageBox.Show("Az adat elmentve.");

this.Close();

}

Törlésnél a program rákérdez a felhasználóra, hogy biztosan törölni szeretné az adatot. Megerősítés után adatbázisból és az adatlistából is kitörli az adatot a program.

//Törlés

private void button3\_Click(object sender, EventArgs e)

{

DialogResult dialogResult = MessageBox.Show("Biztosan törölni szeretné az adatot?", "", MessageBoxButtons.YesNo);

if (dialogResult == DialogResult.Yes)

{

if (Program.askPassword)

{

JelszoKero askPAss = new JelszoKero();

askPAss.ShowDialog();

if (Program.passwordCorrect == false)

{

return;

}

}

Program.conn.Open();

string sqlCommand = "DELETE FROM " + Program.category + " WHERE ID =" + dt.Rows[0].ItemArray[0];

MySqlCommand cmd = new MySqlCommand(sqlCommand, Program.conn);

cmd.CommandType = CommandType.Text;

MySqlDataReader rdr = cmd.ExecuteReader();

rdr.Close();

Program.conn.Close();

Program.writeReport($"Adat [{string.Join(",", kivalasztottSor.data)}] törölve [{Program.category}] kategóriában.");

MessageBox.Show("Az adat törölve.");

this.Close();

}

}

## Jelentés készítése

A program minden egyes tevékenységről jelentést készít például adatszerkesztésnél lementi az eredeti adatot és hogy mire lett változtatva, illetve mikor. Ezeket a jelentés.txt nevű fájlban lementi a program ami ugyan abban a mappában található ahol a program .exe fájla található.

public static void writeReport(string text)

{

FileStream fs = new FileStream(@"jelentés.txt", FileMode.Append);

StreamWriter sw = new StreamWriter(fs, Encoding.UTF8);

sw.WriteLine($"[{DateTime.Now.ToString("yyyy-MM-dd | HH:mm:ss")}]: {text}");

sw.Close();

fs.Close();

}

## Beállítások (Személyreszabás, Jelszó módosítása)

Beállítások ablakban a felhasználónak lehetősége van a program kinézetét változtatni. 4 féle kategóriát lehet változtatni, az ablakok háttérszínét, előtéri színét, szöveg színét illetve a gombok színét.

static private Color backGroundColor;

static private Color foreGroundColor;

static private Color textColor;

static private Color buttonColor;

//Színek

private void button4\_Click(object sender, EventArgs e)

{

ColorDialog backColor = new ColorDialog();

backColor.FullOpen = true;

backColor.Color = Program.backGroundColor;

if (backColor.ShowDialog() == DialogResult.OK)

{

backGroundColor = backColor.Color;

Program.writeReport($"Háttérszín módosítva: {backGroundColor.ToArgb()}");

}

}

private void button5\_Click(object sender, EventArgs e)

{

ColorDialog foreColor = new ColorDialog();

foreColor.FullOpen = true;

foreColor.Color = Program.foreGroundColor;

if (foreColor.ShowDialog() == DialogResult.OK)

{

foreGroundColor = foreColor.Color;

Program.writeReport($"Előtéri szín módosítva: {foreGroundColor.ToArgb()}");

}

}

private void button6\_Click(object sender, EventArgs e)

{

ColorDialog textColorDial = new ColorDialog();

textColorDial.FullOpen = true;

textColorDial.Color = Program.textColor;

if (textColorDial.ShowDialog() == DialogResult.OK)

{

textColor = textColorDial.Color;

Program.writeReport($"Szövegszín módosítva: {textColor.ToArgb()}");

}

}

private void button7\_Click(object sender, EventArgs e)

{

ColorDialog buttonColorDial = new ColorDialog();

buttonColorDial.FullOpen = true;

buttonColorDial.Color = Program.buttonColor;

if (buttonColorDial.ShowDialog() == DialogResult.OK)

{

buttonColor = buttonColorDial.Color;

Program.writeReport($"Gombszín módosítva: {buttonColor.ToArgb()}");

}

}

Beállításokban lehet jelszót változtatni, illetve ki / be lehet kapcsolni, hogy mentéskor a program kérjen-e jelszót.

Illetve az adat exportálásnak az útját meg lehet változtatni.

//Adat Expo Path

private void button3\_Click(object sender, EventArgs e)

{

using (var fbd = new FolderBrowserDialog())

{

fbd.SelectedPath = Program.DataExportPath;

DialogResult newPath = fbd.ShowDialog();

if (newPath == DialogResult.OK && !string.IsNullOrWhiteSpace(fbd.SelectedPath))

{

textBox1.Text = fbd.SelectedPath.ToString();

Program.writeReport($"Adatexportálás helye megváltoztatva: {fbd.SelectedPath.ToString()}");

}

}

}

Mentéskor a beállításokat a program elmenti a config.txt nevű fájlba ami ugyan abban a mappában található ahol a program .exe fájla található. **A jelszót SHA256-os kódolással menti el.**

# Tesztelés

Indítás előtt az adatbázis legyen létrehozva. A következő SQL parancsok szükségesek a felépítéséhez:

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

START TRANSACTION;

SET time\_zone = "+00:00";

CREATE DATABASE IF NOT EXISTS `leltarrendszer` DEFAULT CHARACTER SET utf8 COLLATE utf8\_hungarian\_ci;

USE `leltarrendszer`;

COMMIT;

Illetve az adatbázis eléréséhez a program egy felhasználói fiókot használ. Feltelepítéséhez szükséges SQL parancsok a következők:

CREATE USER 'leltarrendszer'@'localhost' IDENTIFIED BY 'Jelszo123!';

GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, SHOW VIEW ON \*.\* TO 'leltarrendszer'@'localhost';

Ha az adatbázist a program nem tudja elérni, ezt jelszóbekérés után jelezni fogja, majd kilép.

A program legleső indításánál ellenőrízni kell, hogy a config.txt fájlt hiba mentesen létre hozta. Ennek az alap kinézete a következő:

Password=4813494d137e1631bba301d5acab6e7bb7aa74ce1185d456565ef51d737677b2

AskPassword=true

DataExportPath=C:\Users\capta\Desktop\Projects\vizsga\vizsga2\vizsga2\bin\Adatok

BackGroundColor=-986896

ForeGroundColor=-986896

TextColor=-16777216

ButtonColor=-986896

Eltérés lehetséges a DataExportPath-nál. Kódolatlan alap jelszó: „root”

Hozzon létre egy új kategóriát, neve legyen: „géptár”. 2 oszloppal „Leírás”, „Darabszám”. 2 adattal: „Leírás: Fröccsöntő Gép, Darabszám: 2”; „Leírás: Összeszerelő Gép, Darabszám: 3”.

Létrehozásnál a programnak nem szabad átengednie olyan karaktert amit az SQL adatbázis nem tud kezelni, ilyenek példáúl speciális karakterek, szóközök. Illetve a programnak figyelnie kell a számmal kezdődő kategória, oszlop neveket és értesítenie kell a felhasználót, hogy nem kezdőthet számmal. Ugyan ezt a viselkedést kell a programnak kimutatnia meglévő kategória módosításánál. Illetve hogy a változtatásokat, adatokat, kategóriát megfeleően felvitte a program az adatbázisba.

Adatok kiírásánál, kategória kiválasztás után le kell ellenőrízni, hogy minden adatot megfelelően kiír a program.

Keresésnél le kell ellenőrízni, hogy olyan adatot nem jelez ki aminek nincs köze a kereséshez.

Adatbevitelnél le kell ellenőrízni, hogy sikerült-e a rekordot feltöltenie az adatbázisba. Illetve, hogy kimutatja az adatlistába.

Adatmódosításnál meg kell nézni, hogy az adatbázisban is, és az adatlistában is megváltozott az adat.

Adattörlésnél le kell ellenőrízni, hogy a program sikeresen törölte-e az adatbázisból és az adatlistából is az adatot.

Jelentésnél fontos, hogy a program megfelelően létrehozta a jelentés.txt nevű fájlt, illetve hogy az időt, az adat eredeti értéke, és az új értékét írja ki ebben a sorrendben, és lementse, úgy, hogy nem törli az eddigi jelentéseket.

Adatexportálásnál fontos, hogy CSV fájlba, korrupció nélkül mentse ki az adatokkal a jelenleg kiválasztott kategóriából.

Beállításoknál fontos, hogy a program megfelelően megváltoztatta a config.txt fájlt a kért módosításokkal. Szín választásnál le kell ellenőrízni hogy a program újraindítás után megfelelően mutatja ki az új színeket. Illetve jelszó módosításnál, hogy a program megfelelően, SHA256-os kódolással mentette le a jelszót a config.txt fájlba.

# Továbbfejlesztési lehetőségek

## Bejelentkezés

A programot lehetne fejleszteni egy bejelentkezési rendszerrel, ha nagyobb biztonságot szeretnénk és hogy ne lehessen bárkinek jelszót változtatnia, illetve, hogy ne mindig egy fix jelszóval lehessen levédeni. Jelentés szempontjából is előnyösebb lenne mivel ki lehetne íratni, hogy ki csinálta a változtatást.

# Összefoglalás

A program egy leltárrendszer ami folyamatos kapcsolatot tart az adatbázissal működés közben, kategorizál, illetve kimutat külömböző adatokat és azokat a programon belűl lehet módosítani, törölni, létrehozni.

A program főbb funkciói:

* Kategóriák
  + Létrehozás
  + Törlés
  + Módosítás
* Adatkezelés
  + Hozzáadás
  + Törlés
  + Módosítás
  + Keresés
* Jelenetés készítése
* Adatexportálás
* Beállítások
  + Személyreszabás
  + Jelszó módosítás
* Jelszó védelem

# Tartalomjegyzék

[Bevezető 1](#_Toc87472513)

[Felhasználói Dokumentáció 1](#_Toc87472514)

[Jelszó védelem 1](#_Toc87472515)

[Kategóriák 1](#_Toc87472516)

[Adatkezelés 2](#_Toc87472517)

[Jelentés 3](#_Toc87472519)

[Adatexportálás 4](#_Toc87472520)

[Beállítások 4](#_Toc87472521)

[Fejlesztői Dokumentáció 5](#_Toc87472522)

[Bevezető 5](#_Toc87472523)

[Adatbázis 5](#_Toc87472524)

[Kategóriák 5](#_Toc87472525)

[Kategóriák Létrehozása 6](#_Toc87472526)

[Kategóriák Törlése 8](#_Toc87472527)

[Kategóriák Módosítása 9](#_Toc87472528)

[Adatok 10](#_Toc87472529)

[Adatok Hozzáadása 12](#_Toc87472530)

[Adatok Keresése 13](#_Toc87472531)

[Adatexportálás 13](#_Toc87472532)

[Adatok Módosítása, Törlése 14](#_Toc87472533)

[Jelentés készítése 16](#_Toc87472534)

[Beállítások (Személyreszabás, Jelszó módosítása) 16](#_Toc87472535)

[Tesztelés 18](#_Toc87472536)

[Továbbfejlesztési lehetőségek 19](#_Toc87472537)

[Bejelentkezés 19](#_Toc87472538)

[Összefoglalás 20](#_Toc87472539)

[Tartalomjegyzék 21](#_Toc87472540)