Documentatie TSP.NET

Miron Robert-Andrei

March 16, 2020

1 Declaratie

Subsemnatul Miron Robert-Andrei, declar pe propria raspundere ca acest cod nu a fost copiat din internet sau din alte surse. Pentru documentare am folosit urmatoarele surse:

- link-uri:`https://docs.microsoft.com/en-us/dotnet/framework/winforms/`, `https://help.syncfusion.com/windowsforms/overview`, `https://www.geeksforgeeks.org/`, `https://www.youtube.com/`

2 Introducere

Proiectul porneste de la ideea de a avea un sistem care poate manevra(adauga, modifica, cauta, sterge, vizualiza etc) fisierele de tip fotografie si cele de tip video. El consta intr-o aplicatie ce cuprinde doua parti functionale: modelul si API (imbinate intr-un proiect de tip Class Library) si o aplicatie GUI (realizata cu Windows Forms).

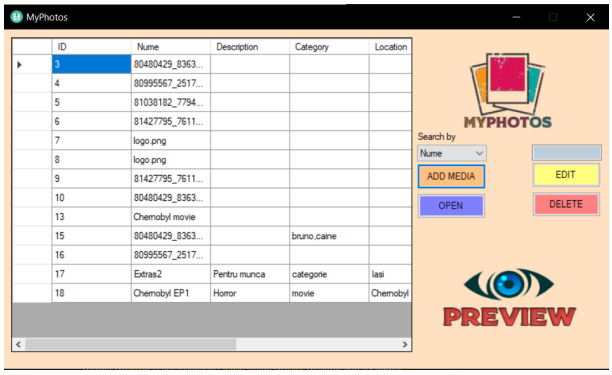
Documentatia aceasta va fi realizata pentru cea de-a doua parte a proiectului, interfata.

Figure 1: Aplicatie

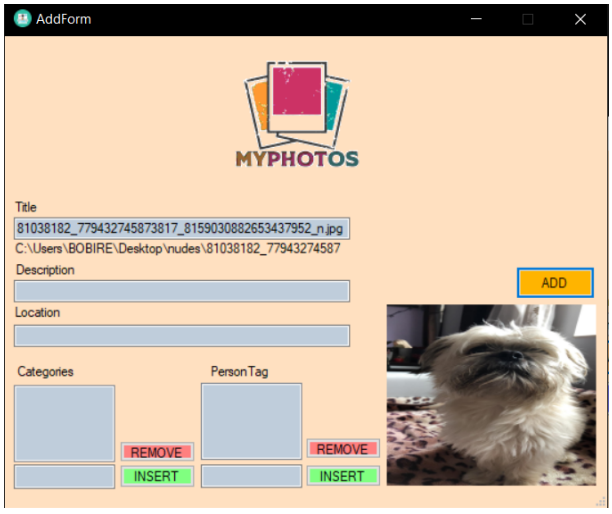
3 Scenarii de utilizare

La deschiderea aplicatiei, se vor observa butoanele de ADD, EDIT, OPEN si DELETE. Pe langa acestea, exista sectiunea de detalii, unde se pot vedea numele, categoria, descrierea etc (toate detaliile fiind preluate din baza de date). Exista, de asemenea, si o sectiune de Preview, ce functioneaza atat pentru imagini, cat si pentru fisiere video.

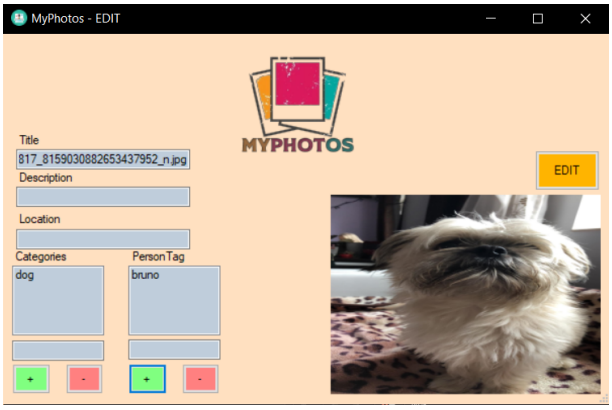
- Ecranul principal la deschiderea aplicatiei, cu functionalitatile descrise mai sus



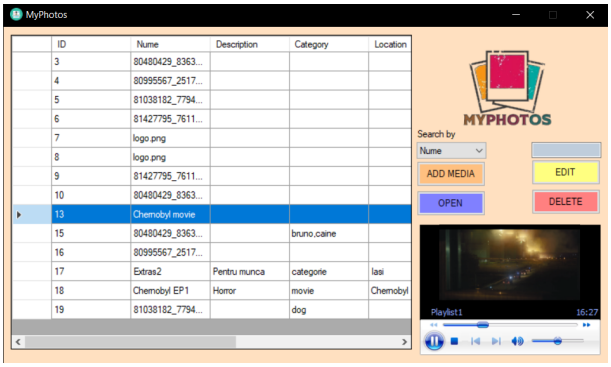
- Functionalitatea butonului Add: se poate observa path-ul fotografiei vizualizata in zona Preview. Exista si butoanele Insert si Remove pentru sectiunile "Categories" si "Person Tag", unde pot fi introduse detalii despre persoana/persoanele care se afla in fisierul introdus, precum si categoria in care se incadreaza.



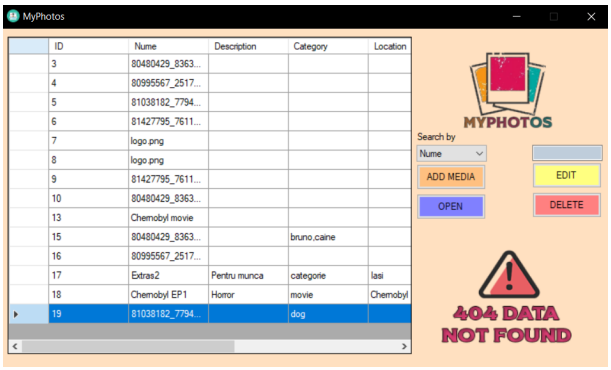
- Functionalitatea butonului Edit: sunt definite aceleasi functionalitati ca la functia de adaugare item, pentru a putea manipula datele introduse fara a interveni in cod; modificarile se realizeaza in mod dinamic.



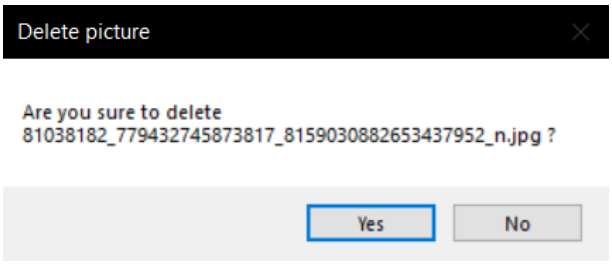
- Demo cu videoclipul afisat in sectiunea Preview.



- In momentul cautarii, fisierele sunt triate automat. In urma cautarii esuate a unui fisier, aplicatia va transmite utilizatorului ca nu mai exista acel item.



- Aplicatia ofera utilizatorului posibilitatea de a anula decizia de a sterge un item.



4 Detalii de implementare

Din sectiunea Toolbox, au fost folosite urmatoarele pentru fiecare Windows Form:

- Label - pentru inserarea textului
- Listbox - pentru afisarea unor liste cu date (categorii si persoane)
- Imagelist - pentru afisarea logo-ului
- WindowsMediaPlayer - pentru derularea videoclipurilor
- Picturebox - pentru sectiunea Preview
- Textbox - pentru citirea textului de la tastatura
- Button - pentru adaugarea functionalitatilor
- DataGridView - pentru afisarea datelor in tabel
- ComboBox - pentru alegerea tipului de filtrare a datelor

Implementarea functiei de filtrare a datelor:
Se preia din ComboBox criteriul de filtrare(nume, path, descriere samd). La fiecare introducere a termenilor de filtrare, DataGrid face update cu datele returnate de catre functia din API.

```
private void filtertext_TextChanged(object sender, EventArgs e)
{
    API api = new API();
    if (textBox1.Text == "")
        UpdateDataGrid();
    else
    {
        var items = api.getDataFiltered(comboBox1.Text, textBox1.Text);
        dataGridView1.DataSource = items.Select(itm => new
        {
            ID = itm.ID_IMG,
            Nume = itm.Name_img,
            Description = itm.Description,
            Category = string.Join(",", itm.Categories.Select(c => c.CategoryName.ToString())),
            Location = itm.Location,
            Persons = string.Join(",", itm.Person.Select(x => x.FirstName.ToString())),
            Date = itm.Data_create,
            Path = itm.Path
        }).ToList();
```

```
public List<PicTable> getDataFiltered(string filterString, string text)
{
    if (filterString == "Nume")
        return dbcontext.PicTableSet.Where(obj => obj.Name_img == text).ToList();
    if (filterString == "Location")
        return dbcontext.PicTableSet.Where(obj => obj.Location == text).ToList();
    if (filterString == "Path")
        return dbcontext.PicTableSet.Where(obj => obj.Path == text).ToList();
    if (filterString == "Descriere")
        return dbcontext.PicTableSet.Where(obj => obj.Description == text).ToList();
    if (filterString == "Person")

        return (from pic in dbcontext.PicTableSet
                where pic.Person.Any(p => p.FirstName == text)
                select pic).ToList();
```

Figure 2: Functia din API care returneaza datele filtrate

La fiecare update al bazei de date, este apelata automat functia Update-DataGrid. In aceasta functie, DataGridView preia datele trimise de catre API, impreuna cu contextul bazei de date.

```
private void UpdateDataGrid()
{
    API api = new API();
    var items = api.GetAllPicsForLoad();
    //dataGridView1.DataSource = items;

    dataGridView1.DataSource = items.Select(itm => new
    {
        ID = itm.ID_IMG,
        Nume = itm.Name_img,
        Description = itm.Description,
        Category = string.Join(",",itm.Categories.Select(c=> c.CategoryName.ToString())),
        Location = itm.Location,
        Persons = string.Join(",", itm.Person.Select(x => x.FirstName.ToString())),
        Date = itm.Data_create,
        Path = itm.Path
    }).ToList();
```