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:

- carti: `https://readthedocs.org/projects/entityframeworkdocs/downloads/pdf/latest/`

- link-uri:`https://docs.microsoft.com/en-us/dotnet/framework/`, `https://www.guru99.com/c-sharp-access-database.html`, `https://profs.info.uaic.ro/~iasimin/csharp_special.htm`, `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 in jurul primei parti a proiectului, si anume modelul, Api si baza de date aferenta.

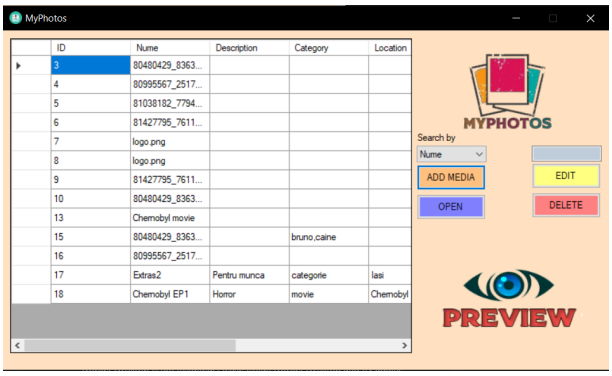

Figure 1: Aplicatie

3 Tehnologiile utilizate

In realizarea **API**-ului sunt implementate functiile din Figura 2. Scopurile lor sunt:

- API() reprezinta constructorul clasei. Se creaza contextul pentru baza de date
- GetAllPicsForLoad() returneaza toate obiectele de tip PicTable
- isInCategories(string) verifica daca un string se afla in baza de date
- existPerson(string) verifica daca persoana exista in baza de date
- AddEntry(...) adauga o poza, precum si detaliile (persoanele aflate in poza, categorie, locatie, descriere etc)
- updateEntry(...) modifica datele unei poze
- RemoveEntry(int) sterge din baza de date
- getDataFiltered(string, string) returneaza toate obiectele pe baza unui filtru

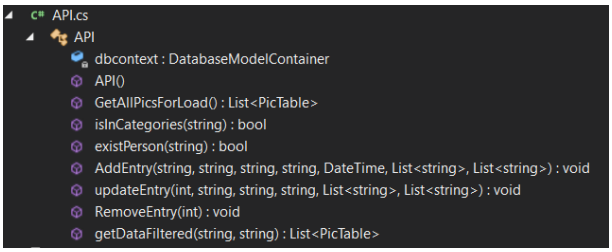

Figure 2: API

Baza de date este formata din trei tabele, dupa cum se poate observa in Figura 3. Exista doua relatii intre tabele mentionate mai sus de tipul many to many (intre PicTable si Persons, precum si PicTable si Categories), pentru a putea adauga dinamic date. Preluarea datelor se realizeaza prin intermediul tehnologiei Linq, utilizand metoda Lambda. Tabela *PicTable* are 6 campuri si este gandita astfel:

- ID_IMG este Primary Key ce se autoincrementeaza
- Name_img este titlul imaginii
- Path este calea catre imagine
- Description este o descriere a imaginii care poate fi introdusa cand este adaugata imaginea, sau ulterior, prin modificare
- Data_create este data la care a fost realizata imaginea
- Location este un loc care, de asemenea, poate fi introdus cand este adaugata imaginea, sau ulterior, prin modificare

Tabela *Person* are 2 campuri:

- ID_Person este Primary Key, cu autoincrement. Prin intermediul acestuia se poate face legatura intre Tabela Person si tabela PicTable
- FirstName este numele persoanei

Tabela *Categories* are 2 campuri:

- ID_Category este Primary Key, cu autoincrement. Face legatura cu tabela PicTable
- CategoryName este numele categoriei

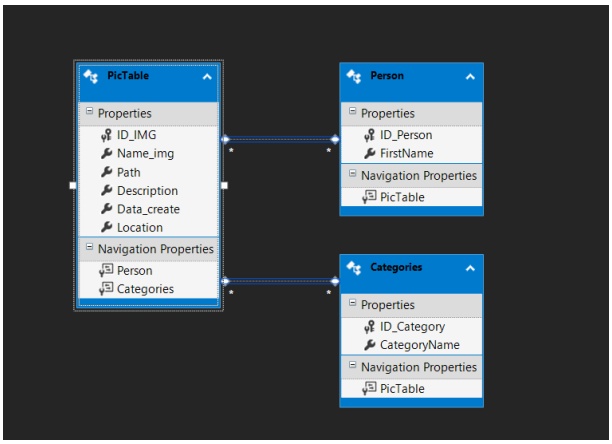

Figure 3: Baza de date

4 Detalii de implementare

In vederea implementarii proiectului, am folosit Entity Framework pentru a putea realiza legatura intre obiectele de tip .NET si baza de date.

Baza de date a fost realizata intr-un item C# de tip ADO.NET Entity Data Model, construita prin metoda Design First. Am conceput tabelele si relatiile dintre ele, apoi, prin functionalitatea incorporata Generate Database from Model, am creat modelul.

```
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();

        //return dbcontext.PicTableSet.Include(p=> p.Person).Where(o => o.Person.Where(p=>p.F
    if (filterString == "Category")
        return (from pic in dbcontext.PicTableSet
                where pic.Categories.Any(c => c.CategoryName== text)
                select pic).ToList();
```

Figure 4: Filtrarea si preluarea datelor