MyPhotos

Grigoraș Alexandru-Ionel

Universitatea Alexandru Ioan Cuza
Facultatea de Informatică
Martie 2020

Cuprins

1 Baza de date

Baza de date pentru mine a reprezentat o singură tabelă(o singură entitate, Figure 1) care să îmi țină datele necesare. Am ales să merg pe această idee deoarece m-am gândit de la început ce inputuri voi avea pentru fiecare poză când va fi adăugată. Nu își are rostul să sparg în mai multe entități atât timp cât Location, Event, Tags, People reprezintă un string. În cazul în care optam să adaug anumite câmpuri legate de fiecare în parte (de exemplu, pentru Location să am Nume, Longitudine, Latitudine etc) atunci ar fi fost o idee bună să sparg Location într-o nouă entitate.

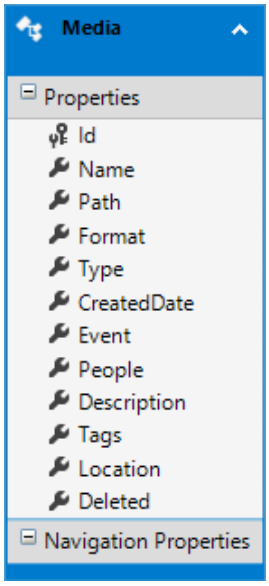

Figure 1: Baza de date

Câmp	Descriere
Id	Are rolul de a identifica unic o resursa
Name	Reprezintă numele pozei/videoclipului, ce poate fi schimbat când se adaugă poza
Path	Reprezintă calea unde se află poza, nu poate fi schimbat
Format	Reprezintă formatul pozei/videoclipului (.jpg, .mp4, ...)
Type	Reprezintă tipul pozei/videoclipului, a fost adăugat pentru a identifica mai rapid dacă este poză sau videoclip
CreatedDate	Reprezintă data când a fost creat/ă poza/videoclipul și este luată automat la adăugare
Event	Un câmp în care utilizatorul poate adăuga date despre evenimentul din poză/videoclip
People	Un câmp în care utilizatorul poate adăuga date despre persoanele din poză/videoclip
Location	Un câmp în care utilizatorul poate adăuga date despre locația din poză/videoclip
Description	Un câmp în care utilizatorul poate adăuga o descriere despre despre poză/videoclip
Tags	Un câmp în care utilizatorul poate adăuga orice dorește pentru identificarea unei poze
Deleted	Reprezintă o valoare între "true" Și "false", ce ne spune dacă imaginea/videoclipul a fost eliminat din aplicație

După ce am creat modelul dorit, am generat baza de date din model. S-a creat automat clasa Media și contextul *MyPhotosContainer*. Aceastea le-am folosit în implementarea "Repository and Unit of Work Patterns". Nu era necesară implementarea lor, dar în cazul în care aș fi dorit să modific la baza de date, să adaug entități, ar fi fost mult mai ușor.

Pentru *Repository Pattern* am creat o clasă generică numită **Repository** ce implementeaza interfața generică **IRepository**. Aici am ales să includ unele funcții generice ce nu țin neapărat de Media, ci pot fi folosite de oricare dintre *ConcreteRepositories*. Trecând mai departe la ConcreteRepositories, am doar unul, acesta a fost numit **MediaRepository** ce implementează interfața IMediaRepository și moștenește **Repository**. Nu a fost nevoie să adaug funcții ce nu țin de generic repository deoarece tot ce am avut nevoie în API, am putut face cu funcțiile generice. Nu ar fi avut rost să adaug funcții doar ca să umplu niște linii de cod.

Trecând mai departe la *UnitOfWork*, avem clasa cu numele respectiv ce implementează interfața **IUnitOfWork**. Pe lângă faptul că în clasa pentru Unit Of Work avem inițializarea repository-ului *MediaRepository* în contrusctor, avem implementată și funcția Complete ce face salvările către baza de date, dar și funcția Dispose deoarece dorim să avem și o funcție pentru eliberarea resurselor.

2 DTO

Pe partea de DTO, am ales să expun un *MediaDTO*. Este doar o singură clasă ce mapează *Media* din baza de date. Aceasta va fi folosită pentru a expune/primii informații către/din exterior. Folosim un Data Transfer Object pentru a transfera date între host și stratul UI. În aplicație DTO-urile vor fi situate în proiectul cu baza de date, in folderul Models/DTO.

3 Declarație

Subsemnatul Grigoraș Alexandru-Ionel declar pe propria raspundere ca acest cod nu a fost copiat din Internet sau din alte surse. Pentru documentare am folosit urmatoarele surse:

→Carti: -

→Link-uri:

- EF6
- EF Model Design First
- LINQ
- Repository + Unit Of Work
- DTO