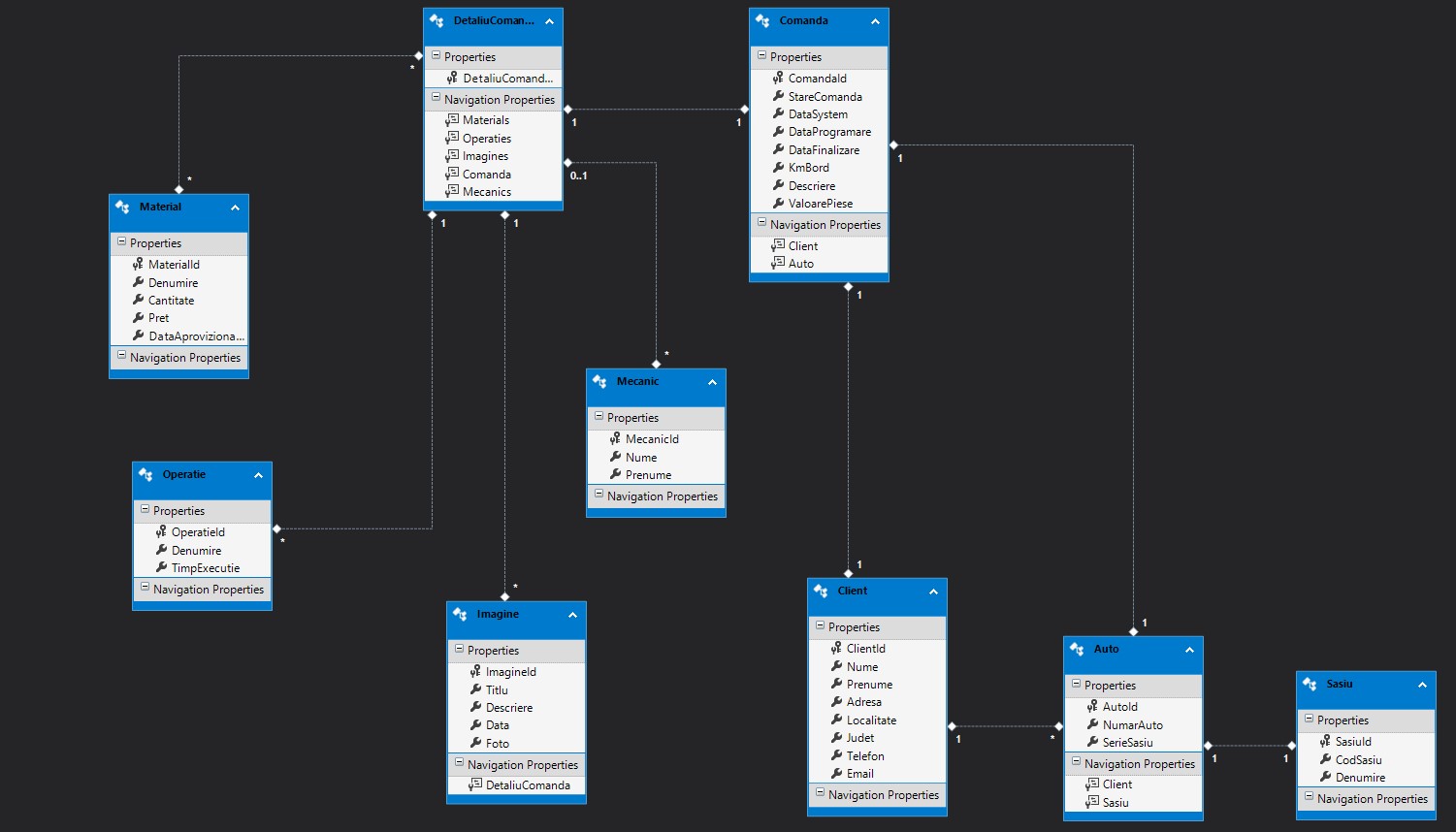
Subsemnatul Loghin L. Alexandru declar pe propria raspundere ca acest cod nu a fost copiat din Internet sau din alte surse.

**ServiceAutoLibrary**

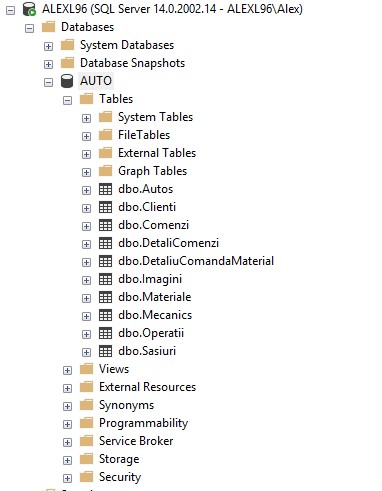
|  |  |  |
| --- | --- | --- |
| **Application Properties** | Assembly name: | ServiceAutoLibrary |
| Default namespace: | ServiceAutoLibrary |
| Target framework: | .NET Framework 4.6.1 |
| Output type: | Class Library |

Am realizat modelul entitatilor urmand cu exactitate pasii din documentatia [EF Model Designer First](https://profs.info.uaic.ro/~iasimin/Laborator%20C%20S%20H/Laborator1-2016.pdf)

La finalul etapei 1 modelul realizat arata ca mai jos sau [aici](https://drive.google.com/open?id=1ExGK0kZYwvjzUte7v-YPQ4JLZzxAYv9k)



Utilizand scriptul general din modele, la final baza de date arata ca mai jos sau [aici](https://drive.google.com/open?id=1hZNu4e1r6PBuobqD1EbYYjpF-JDE9bEh)



**Structura proiectului:**

Pentru logica de design a proiectului am folosit [Repository Pattern](https://medium.com/falafel-software/implement-step-by-step-generic-repository-pattern-in-c-3422b6da43fd) si pentru fiecare model am separat operatiile de citire si de scriere in doua mari categori: Read / Write.

Pentru fiecare entitate avem doua interfete (IRead/IWrite) Repository si doua clase ce implementeaza interfetele respective si contin logica fiecarei metode.

La final, am setat modificatorul de acces pe internal si am adaugat un Service, cu modificator public, ce va contine cate o instanta din fiecare repo.

Iar mai departe, in etape urmatoare, accesul la api si baza de date se va face doar prin acel service.

Interfata IAutoService va cuprinde toate usecase-urile pe care aplicatia le va putea avea, adica pe care utilizatorul le va putea folosi.

La finalul etapei 1 solutia realizata arata ca mai jos sau [aici](https://drive.google.com/open?id=1nb11oAEGCR7OQ_KpfZoeR3WCoAXHrZUw)

