Subsemnatul Costandache Mihai-Andrei declar pe propria răspundere că acest cod nu a fost copiat de pe Internet sau din alte surse. Pentru documentare am folosit următoarele surse: https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application , https://docs.microsoft.com/en-us/dotnet/api/system.reflection?view=netframework-4.6.1

Descriere Proiect 1

Am realizat un proiect de tip Class Library, intitulat CarService, ce expune un API pe care îl voi utiliza în următoarele 3 proiecte. Modelul de programare pe care l-am ales este EF Model Designer First.

Cu ajutorul specificațiilor din pdf-ul ce descrie proiectele, am identificat entitățile și asocierile dintre acestea și am realizat modelul în designer-ul oferit de mediul de dezvoltare VS 2017. Am selectat apoi opțiunea Generate Database From Model, alegând conexiunea la o bază de date denumită AUTO pe care am creat-o în prealabil. Printre altele, s-au generat automat și fișiere conținând cod sursă C#, corespunzător claselor entităților (ex: Auto.cs, Operatie.cs etc.). Nu am modificat aceste fișiere, ci am scris separat cod nou pentru aceste clase, în alte fișiere, cum ar fi Auto.custom.cs sau Operatie.custom.cs. În aceste fișiere am scris cod pentru a construi relații de tip ISA între clasele entităților și o altă clasă, Entitate - pe care am conceput-o pentru a expune o metodă de afișare a unor proprietăți ale entităților, cum ar fi numele unui client, excluzând chei străine sau proprietăți de navigare, dar și în scopul utilizării în tipuri generice. În fișierele de forma <nume_entitate>.custom.cs am scris de asemenea constructori cu parametri, iar în fișierul corespunzător clasei Comanda am scris și o proprietate ValoarePiese care returnează valoarea cumulată a pieselor pe baza materialelor folosite. Pentru realizarea operațiilor CRUD, am folosit pattern-urile Repository și UnitOfWork, clasele corespunzătoare fiind IRepository.cs (interfață), GenericRepository.cs. (implementarea interfeței), IUnitOfWork.cs (interfață), UnitOfWork.cs (implementarea interfeței).

Noutăți

Am folosit pattern-ul Repository pentru a abstractiza operațiile CRUD, cu ajutorul unor repository-uri generice de entități și pattern-ul UnitOfWork pentru a opera pe aceste repository-uri în același context. IRepository prezintă o serie de operații CRUD, iar GenericRepository implementează interfața, primind în constructor contextul pe care se lucrează. Interfața IUnitOfWork este o interfață ce moștenește IDisposable și prezintă proprietăți de acces la repository-uri generice dar și o metodă Save(), UnitOfWork este clasa care implementează interfața, ce primește în constructor un context și îl folosește la instanțierea repository-urilor. De asemenea, UnitOfWork salvează modificările prin metoda Save() și implementează Dispose().

Alte noutăți ar fi păstrarea tuturor proprietăților de navigare și a cheilor străine sugerate de mediul de dezvoltare și calcularea valorii pieselor pentru o comandă printr-o proprietate din clasa Comanda.custom.cs, fără a avea în baza de date o coloană în tabelul Comenzi pentru valoarea pieselor.