Documentație proiect 1 CarService

Subsemnata Bîrsan Ioana, declar pe proprie răspundere că acest cod nu a fost copiat din Internet sau din alte surse.

Bibliografie:

- https://blog.devart.com/set-identity-and-computed-properties-in-entity-framework-without-triggers.html
- https://stackoverflow.com/questions/25894587/how-to-update-record-using-entity-framework-6
- https://stackoverflow.com/questions/35799017/adding-attributes-to-ef6-generated-entity-classes
- http://alexwolfthoughts.com/adding-validation-metadata-to-entity-framework-generated-classes/
- https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/database-first-development/enhancing-data-validation
- https://dev.to/kylegalbraith/getting-familiar-with-the-awesome-repository-pattern--1ao3
- https://deviq.com/repository-pattern/

Noutăți:

Deoarece aveam nevoie să adaug reguli de validare folosind atribute la nivelul proprietăților din clasele parțiale generate, iar acestea s-ar fi rescris la următoarea regenerare, am folosit următoarea abordare:

- pentru fiecare clasă parțială generată (e.g. `Auto`), am creat o clasă nouă (e.g. `AutoMetadata`), în care am adăugat reguli de validare la nivel de proprietate
- pentru fiecare clasă parțială generată am creat o nouă clasă parțială (e.g. `Auto`) și am utilizat atributul MetadataType (e.g. [`MetadataType(typeof(AutoMetadata))`])

Aceste reguli sunt acum impuse în proiect și nu vor fi șterse în momentul în care se regenerează clasele. Clasele create se află în folderul **Models**.

Am implementat șablonul Repository pentru a oferi un nivel de abstractizare între nivelul de persistență (baza de date AUTO) și nivelul de business (`ICarService`, `CarService`). Am creat o interfață generică `IRepository` în care expun următoarele metode:

```
public interface IRepository<T>
      {
      T GetById(int id);
      IReadOnlyList<T> GetAll();
      void Create(T item);
      void Delete(int id);
      void Update(T item);
      bool Exists(int id);
      void SaveChanges();
      }
```

Pentru fiecare model, am implementat această interfață, folosind ca dependență contextul generat `CarServiceModelContainer`. Unde a fost cazul am expus și alte metode necesare pentru logica aplicației. Nivelul acesta se regăsește în folderul **Repository**.

Nivelul de logică al aplicației, care expune API-ul cerut, se află în folderul **Service**. Am creat o interfață: `ICarService` și o clasă care o implementează: `CarService`. Dependențele folosite sunt cele din nivelul de repository.

Pentru testare, am creat în cadrul aceleiași soluții un proiect de tip Console App (TestCarService) căruia i-am adăugat referință către proiectul de tip Class Library (CarService). În arhivă voi adăuga doar fișierul numit `Program.cs`.