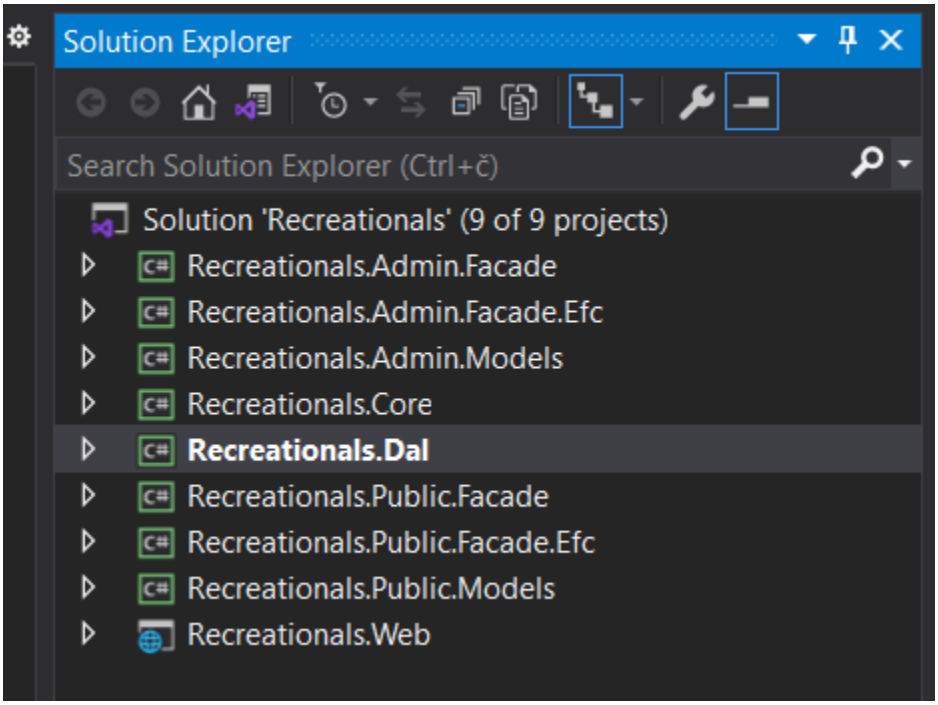
Refactoring - ***Rekreativci***

Code refactoring je od velikog značaja za cijeli životni ciklus naše aplikacije. Svi kada pravimo aplikaciju težimo prvobitno samo prema funkcionalnostima, ne obazirajući se na kvalitetu koda i ne razmišljajući o budućnosti i održavanju tog samog koda.

Tu dolazi code refactoring, mi želimo da nam aplikacija koju smo napravili i koju planiramo nadograđivati bude dugoročna i jednostavna za ispravke, modifikaciju i naposlijetku razumijevanje.

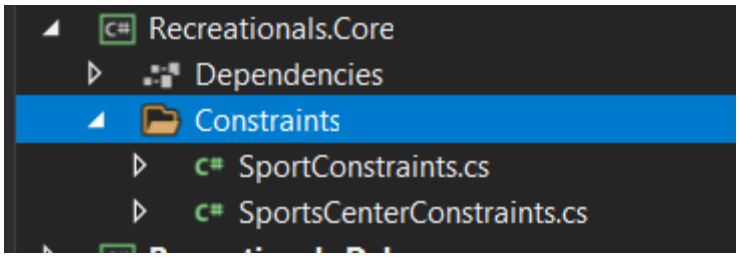
Prvo što ćemo pokazati je raščlanjivanje samih dijelova koda, na sljedećoj slici možemo vidjeti kako nam je pomogao patern koji smo i ranije radili, to je Facade patern koji se često koristi u ovu svrhu kako smo i vidjeli na mnogim primjerima na Internetu.



Možemo vidjeti koliko je lakše prolaziti kroz cijeli projekat i tačno po imenima znamo čemu služi koji folder. Ovo će u budućnosti mnogo pomoći pri snalaženju, popravljanju i traženju potencijalnih grešaka kao i nadogradnji same aplikacije.

Znamo da je jedan od najvećih code smells-a ponavljanje koda, iako je naša aplikacija vrlo jednostavna, htjeli smo ipak pokazati kako bi se ovo moglo ostvariti. Svaka forma bi morala imati neke zahtjeve tj. ograničenja, npr. na dužinu username-a, kompleksnosti passworda, ...

Znamo koliko je naporno stalno kucati jedno po jedno ograničenje pa smo uradili sljedeće...



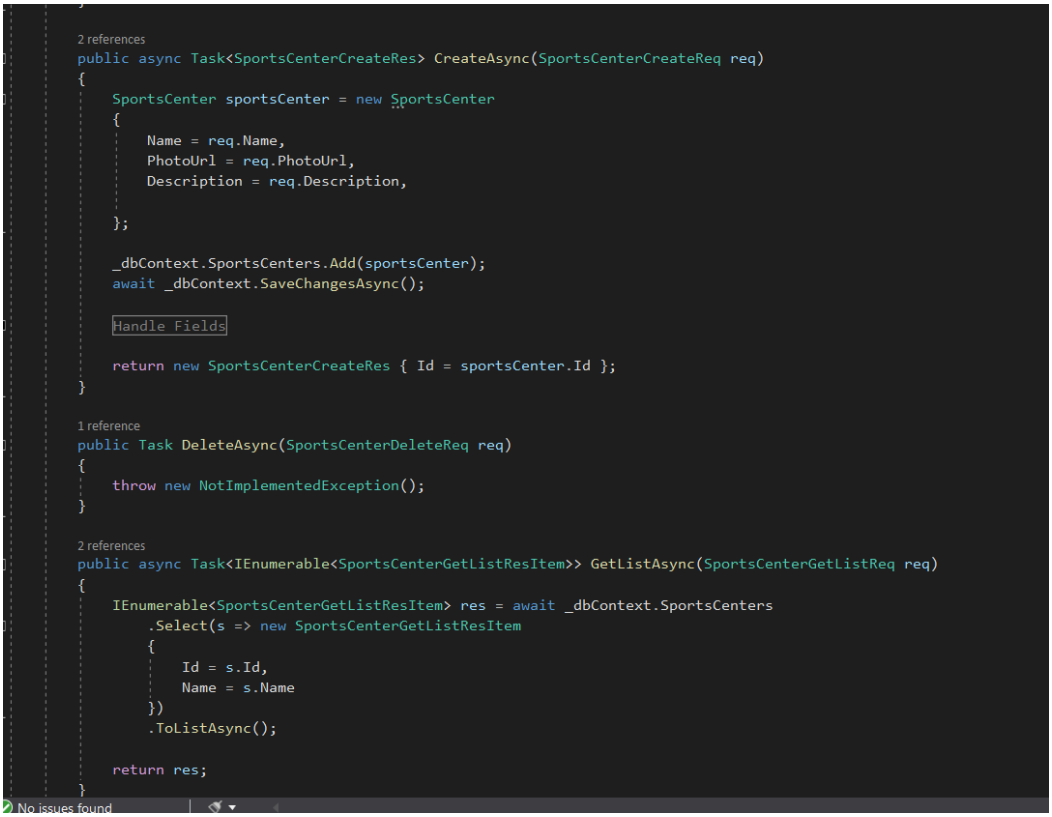
U ovom folderu držimo sve zahtjeve, tako da kada je npr. potrebno promijeniti jedan zahtjev to lagano možemo uraditi na jednom mjestu i to će se automatski odraziti i na ostala.

U nastavku slijede naši zahtjevi koje smo napisali za modele Sport i SportskiCentar.

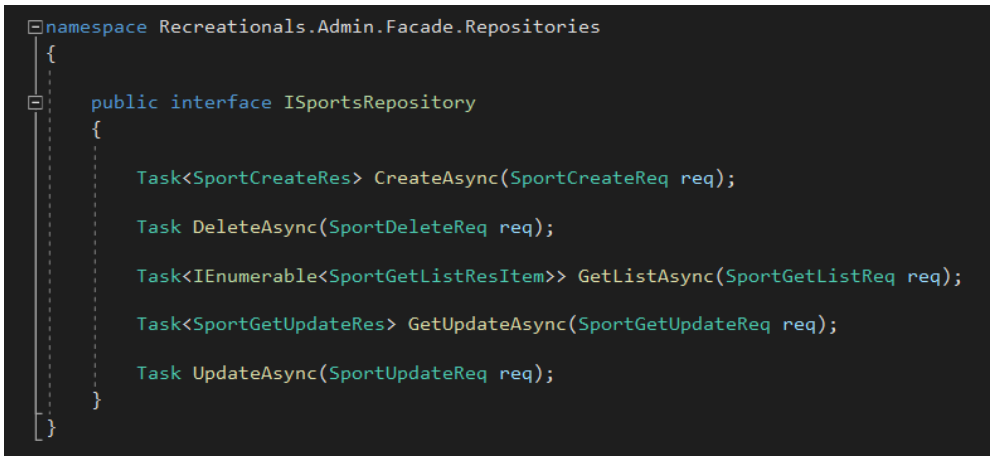
```
tionals.Core                                    Recreationals.Core.Const
namespace Recreationals.Core.Constraints
{
    6 references
    public class SportConstraints
    {
        public const int MinNameLength = 2;
        public const int MaxNameLength = 100;

        public const int MinPhotoUrlLength = 2;
        public const int MaxPhotoUrlLength = 1000;
    }
}
```

```
Recreationals.Core                              Recreationals.Core.
1   namespace Recreationals.Core.Constraints
2   {
3       public class SportsCenterConstraints
4       {
5           public const int MinNameLength = 2;
6           public const int MaxNameLength = 100;
7
8           public const int MinDescriptionLength = 2;
9           public const int MaxDescriptionLength = 1000;
10
11          public const int MinAddressLength = 2;
12          public const int MaxAddressLength = 200;
13
14          public const int MinPhotoUrlLength = 2;
15          public const int MaxPhotoUrlLength = 1000;
16
17          public const int MinFacebookUrlLength = 2;
18          public const int MaxFacebookUrlLength = 200;
19
20          public const int MinWebsiteUrlLength = 2;
21          public const int MaxWebsiteUrlLength = 200;
22
23          public const int MinPhoneNumberLength = 2;
24          public const int MaxPhoneNumberLength = 200;
25      }
26  }
```

Također, popravljali smo dosta kod jer smo na prijašnjim „programerskim“ predmetima navikli pisati duge funkcije / metode. U ovom projektu smo se držali pravila da su najbolje funkcije / metode one koje su duge svega par linija koda. Na sljedećoj slici možemo vidjeti kratki isječak koda gdje se vidi maloprije spomenuto.



Čitajući knjigu o clean code-u na predmetu RPR, naučili smo da je dobra praksa također praviti interfejse kako bi se lakše shvatila poenta same klase, te samim tim i lakše implementirala. Mi smo to uradili (*iako na već jednostavnom sistemu*) tako što smo napravili dva interfejsa za Sport i SportskiCentar.



```
namespace Recreationals.Admin.Facade.Repositories
{

    public interface ISportsRepository
    {

        Task<SportCreateRes> CreateAsync(SportCreateReq req);

        Task DeleteAsync(SportDeleteReq req);

        Task<IEnumerable<SportGetListResItem>> GetListAsync(SportGetListReq req);

        Task<SportGetUpdateRes> GetUpdateAsync(SportGetUpdateReq req);

        Task UpdateAsync(SportUpdateReq req);
    }
}
```

I na kraju možemo reći da smo pazili na samo imenovanje cijelog projekta i svih njegovih gradivnih jednica, jer još jedan vrlo bitan aspect čistoće i ljepote samog koda je pravilno i **konzistentno** imenovanje svega u sistemu, gledajući već sve prijašnje slike može se vidjeti da to naš sistem poštuje.