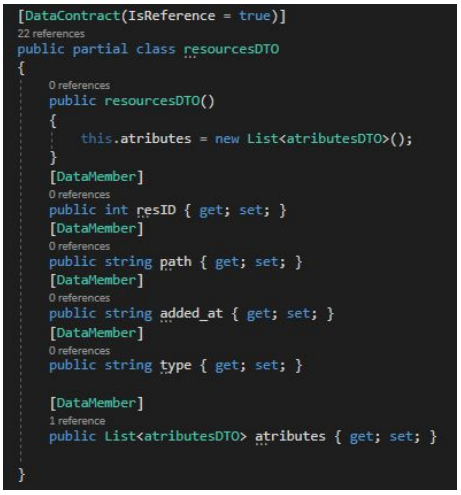
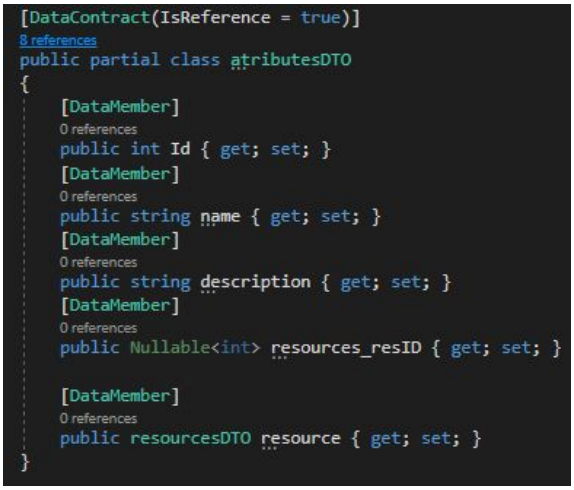
Am creat 3 contracte, doua DataContract : resourcesDTO si atributesDTO si un ServiceContract : InterfaceWCF. Am folosit DataContract pentru a putea abstractiza datele oferite de serviciu, luate din baza de date, astfel incat sa le pot trimite clientului. Astfel clientul si serviciul nu trebuie sa aiba aceleasi tipuri de date ci doar trebuie sa aiba aceleasi contracte.



```
[DataContract(IsReference = true)]
22 references
public partial class resourcesDTO
{
    0 references
    public resourcesDTO()
    {
        this.atributes = new List<atributesDTO>();
    }
    [DataMember]
    0 references
    public int resID { get; set; }
    [DataMember]
    0 references
    public string path { get; set; }
    [DataMember]
    0 references
    public string added_at { get; set; }
    [DataMember]
    0 references
    public string type { get; set; }

    [DataMember]
    1 reference
    public List<atributesDTO> atributes { get; set; }

}
```



```
[DataContract(IsReference = true)]
8 references
public partial class atributesDTO
{
    [DataMember]
    0 references
    public int Id { get; set; }
    [DataMember]
    0 references
    public string name { get; set; }
    [DataMember]
    0 references
    public string description { get; set; }
    [DataMember]
    0 references
    public Nullable<int> resources_resID { get; set; }

    [DataMember]
    0 references
    public resourcesDTO resource { get; set; }
}
```

ServiceContractul (InterfaceWCF) defineste o interfata formata din functii identice cu cele definite in api.

In Photos.cs se afla implementarea interfetei. Am folosit AutoMapper pentru a face o mapare eficeinta a resourcesDTO cu resources si atributesDTO cu atributes. Adica am mapat intre contractul de date si clasele din baza de date.

```
public List<resourcesDTO> getAllResources()
{

    var getAll = api.getAllResources();
    var resDTO = new List<resourcesDTO>();
    resDTO = iMapper.Map<List<resources> , List<resourcesDTO>>(getAll);
    return resDTO;

}
1 reference
public List<resourcesDTO> getAllPhotos()
{
    var getAll = api.getAllResources();
    List<resourcesDTO> resDTO = new List<resourcesDTO>();
    resDTO = iMapper.Map<List<resources>, List<resourcesDTO>>(getAll, resDTO);
    return resDTO;
}
```

Implementarea functiilor din interfata presupune maparea datelor si apoi apelarea functiilor din api si returnarea datelor returnate de acestea.

```
[ServiceContract]
1 reference
public interface InterfaceWCF
{
    [OperationContract]
    1 reference
    int test();

    [OperationContract]
    1 reference
    List<resourcesDTO> getAllResources();

    [OperationContract]
    1 reference
    List<resourcesDTO> getAllPhotos();
    [OperationContract]
    1 reference
    List<resourcesDTO> getAllVideos();
    [OperationContract]
    1 reference
    List<string> getAllAtributes();
    [OperationContract]
    1 reference
    List<atributesDTO> getAllAtribtuesForResource(string path);

    [OperationContract]
    1 reference
    List<resourcesDTO> getResourcesByQueryAtribute(string atr, string query);
    [OperationContract]
    1 reference
    void saveResources(string path, string type, List<string> atr);
    [OperationContract]
    1 reference
    void updateResource(string path, List<string> atr);
    [OperationContract]
    1 reference
    void deleteResources(List<string> paths);
    [OperationContract]
    1 reference
    void deleteAtribute(List<string> names);
}
```