Laborator WCF – 24.03.2020

1. Problema prezentata la curs. Scop: urmarirea etapelor in realizarea proiectului.
2. WCF si EF – prezentat in continuare.

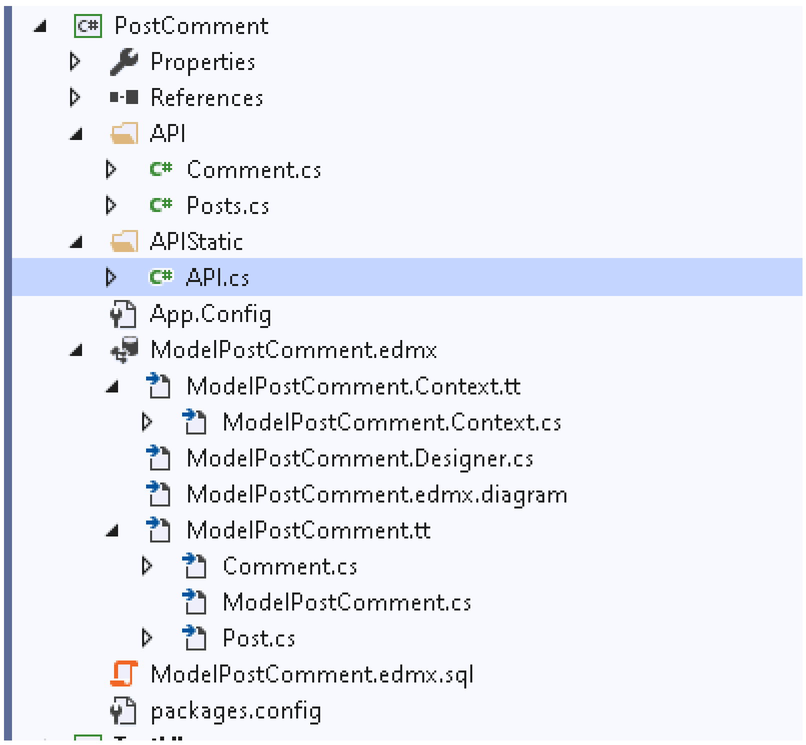
Etapa 1. Construire baza de date. Modelul ales este “Model Designer First”. Proiect **ClassLibrary** (.NET Framework).

Se lanseaza Visual Studio in mod **Administrator**.

Observatie
Daca intampinati probleme revedeti laboratorul “*EF Model Designer First*”.

Eu am numit proiectul “PostComment”.

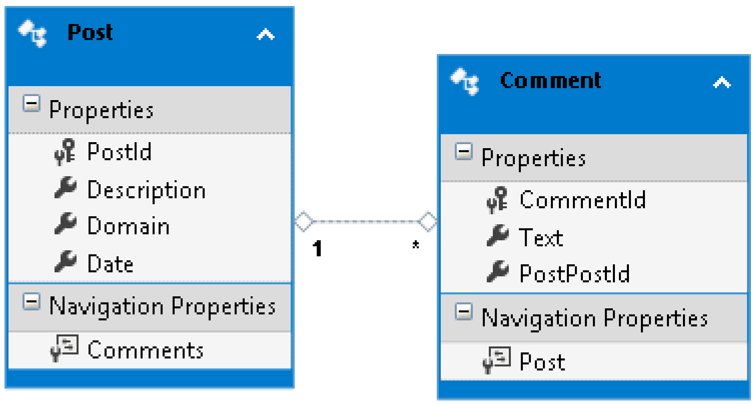
In Visual Studio proiectul PostComment arata astfel:



```
PostComment
  Properties
  References
  API
    Comment.cs
    Posts.cs
  APIStatic
    API.cs
  App.Config
  ModelPostComment.edmx
    ModelPostComment.Context.tt
      ModelPostComment.Context.cs
    ModelPostComment.Designer.cs
    ModelPostComment.edmx.diagram
    ModelPostComment.tt
      Comment.cs
      ModelPostComment.cs
      Post.cs
  ModelPostComment.edmx.sql
  packages.config
```

La proiect am adaugat doua foldere: API si APIStatic. In API am scris metode ale instantei ce lucreaza cu baza de date, in APIStatic am scris metode statice (fisier API.cs) ce lucreaza cu baza de date.

Rezultatul este:



Observati spatiul de nume “**PostComment**”.

Clasele generate au fost modificate astfel:

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated from a template.
//
//     Manual changes to this file may cause unexpected behavior in your application.
//     Manual changes to this file will be overwritten if the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace PostComment
{
    using System;
    using System.Data.Entity;
    using System.Data.Entity.Infrastructure;

    internal partial class ModelPostCommentContainer : DbContext
    {
        public ModelPostCommentContainer()
            : base("name=ModelPostCommentContainer")
        {
              // Inhibare: lazy loading si creare proxy dinamic.
            Configuration.LazyLoadingEnabled = false;
            Configuration.ProxyCreationEnabled = false;
        }

        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            throw new UnintentionalCodeFirstException();
        }

        public virtual DbSet<Post> Posts { get; set; }
        public virtual DbSet<Comment> Comments { get; set; }
    }
}
```

Clasa Post. Adnotare cu DataContract si DataMember.

Utilizarea atributelor **DataContract** si **DataMember**. Necesar pentru declararea contractului de date in Windows Communication Foundation.

[**DataContract(IsReference = true)**] previne referinte circulare in WCF intre ***Post*** si ***Comment***.

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated from a template.
//
//     Manual changes to this file may cause unexpected behavior in your application.
//     Manual changes to this file will be overwritten if the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace PostComment
{
    using System;
    using System.Collections.Generic;
    using System.Runtime.Serialization;

    [DataContract(IsReference=true)]
    public partial class Post
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
              "CA2214:DoNotCallOverridableMethodsInConstructors")]
        public Post()
        {
            this.Comments = new HashSet<Comment>();
        }

        [DataMember]
        public int PostId { get; set; }
        [DataMember]
        public string Description { get; set; }
        [DataMember]
        public string Domain { get; set; }
        [DataMember]
        public System.DateTime Date { get; set; }

        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
               "CA2227:CollectionPropertiesShouldBeReadOnly")]
        [DataMember]
        public virtual ICollection<Comment> Comments { get; set; }
    }
}
```

Clasa Comment

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated from a template.
//
//     Manual changes to this file may cause unexpected behavior in your application.
//     Manual changes to this file will be overwritten if the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace PostComment
{
    using System;
    using System.Collections.Generic;
    using System.Runtime.Serialization;

    [DataContract(IsReference=true)]
    public partial class Comment
    {
        [DataMember]
        public int CommentId { get; set; }
        [DataMember]
        public string Text { get; set; }
        [DataMember]
        public int PostPostId { get; set; }
        [DataMember]
        public virtual Post Post { get; set; }
    }
}
```

Cream si adaugam la proiect alte clase "partial" *Post* si *Comment* in care scriem codul pentru metode (API). La compilare din cele doua clase se creaza una singura. In acest fel nu pierdem codul scris in situatia cand regeneram baza de date din model.

Clasele Post si Comment au fost completate astfel:

Clasa Post

```
using System.Collections.Generic;
using System.Linq;
using System.Data.Entity;

namespace PostComment
{
    public partial class Post
    {
        public bool AddPost()
        {
            using (ModelPostCommentContainer ctx = new ModelPostCommentContainer())
            {
                bool bResult = false;
                if (this.PostId == 0)
                {
                    var it = ctx.Entry<Post>(this).State = EntityState.Added;
                    ctx.SaveChanges();
                    bResult = true;
                }
                return bResult;
            }
        }

        public Post UpdatePost(Post newPost)
        {
            using (ModelPostCommentContainer ctx = new ModelPostCommentContainer())
            {
                Post oldPost = ctx.Posts.Find(newPost.PostId);
                if (oldPost == null) // nu exista in bd
                {
                    return null;
                }
                oldPost.Description = newPost.Description;
                oldPost.Domain = newPost.Domain;
                oldPost.Date = newPost.Date;
                ctx.SaveChanges();
                return oldPost;
            }
        }

        public int DeletePost(int id)
        {
            using (ModelPostCommentContainer ctx = new ModelPostCommentContainer())
            {
                return ctx.Database.ExecuteSqlCommand("Delete From Post where postid =
                      @p0", id);
            }
        }

        public Post GetPostById(int id)
        {
            using (ModelPostCommentContainer ctx = new ModelPostCommentContainer())
            {
                var items = from p in ctx.Posts where (p.PostId == id) select p;
                if (items != null)
                    return items.Include(c => c.Comments).SingleOrDefault();
                return null; // trebuie verificat in apelant
            }
        }

        public List<Post> GetAllPosts()
        {
            using (ModelPostCommentContainer ctx = new ModelPostCommentContainer())
            {
                return ctx.Posts.Include("Comments").ToList<Post>();
                // Obligatoriu de verificat in apelant rezultatul primit.
            }
        }

    }
}
```

Classa Comment

```
using System.Linq;
using System.Data.Entity;

namespace PostComment
{
    public partial class Comment
    {

        public bool AddComment()
        {
            using (ModelPostCommentContainer ctx = new ModelPostCommentContainer())
            {
                bool bResult = false;
                if (this == null || this.PostPostId == 0)
                    return bResult;
                if (this.CommentId == 0)
                {
                    ctx.Entry<Comment>(this).State = EntityState.Added;
                    Post p = ctx.Posts.Find(this.PostPostId);
                    ctx.Entry<Post>(p).State = EntityState.Unchanged;
                    ctx.SaveChanges();
                    bResult = true;
                }
                return bResult;
            }
        }

        public Comment UpdateComment(Comment newComment)
        {
            using (ModelPostCommentContainer ctx = new ModelPostCommentContainer())
            {
                Comment oldComment = ctx.Comments.Find(newComment.CommentId);
                // Deoarece parametrul este un Comment ar trebui verificata fiecare
                // proprietate din newComment daca are valoare atribuita si
               // daca valoarea este diferita de cea din bd.
                // Acest lucru il fac numai la modificarea asocierii.
                if (newComment.Text != null)
                    oldComment.Text = newComment.Text;
                if ((oldComment.PostPostId != newComment.PostPostId)
                           && (newComment.PostPostId != 0))
                {
                    oldComment.PostPostId = newComment.PostPostId;
                }
                ctx.SaveChanges();
                return oldComment;
            }
        }

        public Comment GetCommentById(int id)
        {
            using (ModelPostCommentContainer ctx = new ModelPostCommentContainer())
            {
                var items = from c in ctx.Comments where (c.CommentId == id) select c;
                return items.Include(p => p.Post).SingleOrDefault();
            }
        }
    }
}
```

Metodele de mai sus sunt metode ale instantei.

Varianta cu metode statice. Adaugam la proiect clasa statica numita **API.**

```
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;

namespace PostComment.APIStatic
{
    public static class API
    {
        public static bool AddPost(Post post)
        {
            using (ModelPostCommentContainer ctx = new ModelPostCommentContainer())
            {
                bool bResult = false;
                if (post.PostId == 0)
                {
                    var it = ctx.Entry<Post>(post).State = EntityState.Added;
                    ctx.SaveChanges();
                    bResult = true;
                }
                return bResult;
            }
        }

        public static Post UpdatePost(Post newPost)
        {
            using (ModelPostCommentContainer ctx = new ModelPostCommentContainer())
            {
                // Ce e in bd. PK nu poate fi modificata
                Post oldPost = ctx.Posts.Find(newPost.PostId);
                if (oldPost == null) // nu exista in bd
                {
                    return null;
                }
                oldPost.Description = newPost.Description;
                oldPost.Domain = newPost.Domain;
                oldPost.Date = newPost.Date;
                ctx.SaveChanges();
                return oldPost;
            }
        }

        public static int DeletePost(int id)
        {
            using (ModelPostCommentContainer ctx = new ModelPostCommentContainer())
            {
                return ctx.Database.ExecuteSqlCommand("Delete From Post where postid =
                            @p0", id);
            }
        }

        /// <summary>
        /// Returnez un Post si toate Comment-urile asociate lui
        /// </summary>
        /// <param name="id"></param>
        /// <returns></returns>
        public static Post GetPostById(int id)
        {
            using (ModelPostCommentContainer ctx = new ModelPostCommentContainer())
            {
                var items = from p in ctx.Posts where (p.PostId == id) select p;
                if (items != null)
                    return items.Include(c => c.Comments).SingleOrDefault();
                return null;
            }
        }

        /// <summary>
        /// Returnez toate Post-urile si Comment-urile corespunzatoare
        /// </summary>
        /// <returns></returns>
        public static List<Post> GetAllPosts()
        {
            using (ModelPostCommentContainer ctx = new ModelPostCommentContainer())
            {
                return ctx.Posts.Include("Comments").ToList<Post>();
            }
        }

        // Comment table
        public static bool AddComment(Comment comment)
        {
            using (ModelPostCommentContainer ctx = new ModelPostCommentContainer())
            {
                bool bResult = false;
                if (comment == null || comment.PostPostId == 0)
                    return bResult;
                if (comment.CommentId == 0)
                {
                    ctx.Entry<Comment>(comment).State = EntityState.Added;
                    Post p = ctx.Posts.Find(comment.PostPostId);
                    ctx.Entry<Post>(p).State = EntityState.Unchanged;
                    ctx.SaveChanges();
                    bResult = true;
                }
                return bResult;
            }
        }

        public static Comment UpdateComment(Comment newComment)
        {
            using (ModelPostCommentContainer ctx = new ModelPostCommentContainer())
            {
                Comment oldComment = ctx.Comments.Find(newComment.CommentId);
                if (newComment.Text != null)
                    oldComment.Text = newComment.Text;
                if ((oldComment.PostPostId != newComment.PostPostId)
                            && (newComment.PostPostId != 0))
                {
                    oldComment.PostPostId = newComment.PostPostId;
                }
                ctx.SaveChanges();
                return oldComment;
            }
        }

        public static Comment GetCommentById(int id)
        {
            using (ModelPostCommentContainer ctx = new ModelPostCommentContainer())
            {
                var items = from c in ctx.Comments where (c.CommentId == id) select c;
                return items.Include(p => p.Post).SingleOrDefault();
            }
        }
    }
}
```

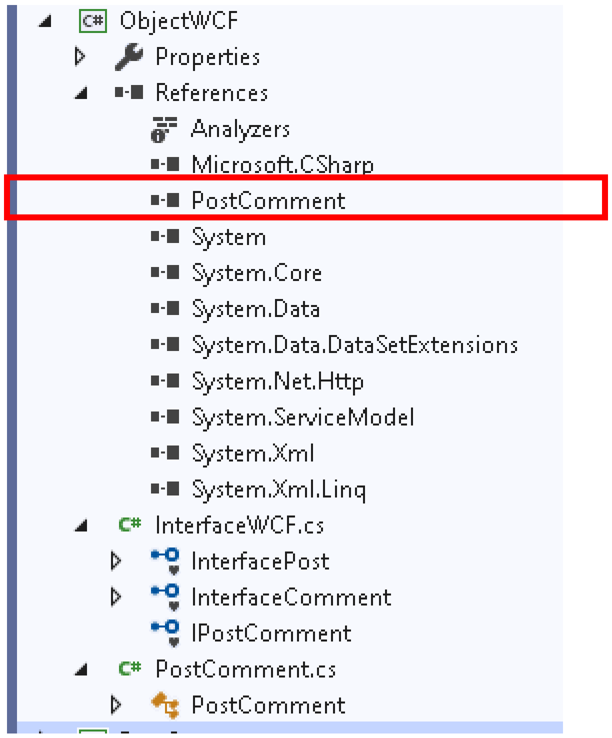
Am terminat cu primul proiect.

In aceeasi solutie cream urmatorul proiect.

Etapa 2. Proiect ClassLibrary (.NET Framework) si l-am numit ObjectWCF. Obiectele din serviciu WCF.

Construire obiecte pentru WCF. Interfetele. Nu sunt luate in considerare metodele statice. Sunt comentate, partial.

In VS proiectul arata astfel. Vedeti referinta adaugata.


```
ObjectWCF
  Properties
  References
    Analyzers
    Microsoft.CSharp
    PostComment
    System
    System.Core
    System.Data
    System.Data.DataSetExtensions
    System.Net.Http
    System.ServiceModel
    System.Xml
    System.Xml.Linq
  InterfaceWCF.cs
    InterfacePost
    InterfaceComment
    IPostComment
  PostComment.cs
    PostComment
```

Se adauga **referinta** la primul proiect si apoi directiva “using PostComment”. Atentie la ce spatiu de nume folositi. Acest cod se gaseste in InterfaceWCF.cs

```
using System.Collections.Generic;

using System.ServiceModel;
using PostComment;

namespace ObjectWCF
{
    [ServiceContract]
    interface InterfacePost
    {
        [OperationContract]
        bool AddPost(Post post);

        [OperationContract]
        Post UpdatePost(Post post);
        [OperationContract]
        int DeletePost(int id);

        [OperationContract]
        Post GetPostById(int id);

        [OperationContract]
        List<Post> GetPosts();
    }

    [ServiceContract]
    interface InterfaceComment
    {
        [OperationContract]
        bool AddComment(Comment comment);

        [OperationContract]
        Comment UpdateComment(Comment newComment);

        [OperationContract]
        Comment GetCommentById(int id);
    }

    [ServiceContract]
    interface IPostComment: InterfacePost, InterfaceComment
    {
    }
}
```

Implementare. Metodele statice nu sunt apelate. Sunt comentate. Codul se gaseste in PostComment.cs

```
using System;
using System.Collections.Generic;
using PostComment;
using PostComment.APIStatic;

namespace ObjectWCF
{
    public class PostComment : IPostComment
    {
        bool InterfaceComment.AddComment(Comment comment)
        {
            return comment.AddComment();
        }

        bool InterfacePost.AddPost(Post post)
        {
            return post.AddPost();
            //return API.AddPost(post);
        }

        int InterfacePost.DeletePost(int id)
        {
            Post post = new Post();
            return post.DeletePost(id);
            // static
            //return API.DeletePost(id);
        }

        Comment InterfaceComment.GetCommentById(int id)
        {
            Comment comment = new Comment();
            return comment.GetCommentById(id);
        }

        Post InterfacePost.GetPostById(int id)
        {
            // E nevoie de ac instanta. Metodele din API sunt metode ale instantei.
            Post post = new Post();
            // Mesaj ce apare in server CUI. Nu e necesar.
            Console.WriteLine("GetPostById. Id = {0}", id);
            post = post.GetPostById(id); // Neclar acest cod.
            Console.WriteLine("Post returnat. Id = {0} , Description = {1}",
                     post.PostId, post.Description);
            return post;
        }

        List<Post> InterfacePost.GetPosts()
        {
            Post post = new Post();
            return post.GetAllPosts();
        }

        Comment InterfaceComment.UpdateComment(Comment newComment)
        {
            return newComment.UpdateComment(newComment);
        }

        Post InterfacePost.UpdatePost(Post post)
        {
            return post.UpdatePost(post);
        }
    }
}
```

In cadrul aceleasi solutii cream urmatorul proiect.

Etapa 3. Host pentru serviciu. App Console (.NET Framework) -CUI. Numele este HostWCF.

Proiectul adauga referinta la proiectul **ObjectWCF** – proiectul din etapa a doua a.
Adaugati refrinte la **System.ServiceModel** si **System.ServiceModel.Description**.

```
using System;

using System.ServiceModel;
using ObjectWCF;
using System.ServiceModel.Description;

namespace HostWCF
```

```
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Lansare server WCF...");
            ServiceHost host = new ServiceHost(typeof(PostComment),
                     new Uri("http://localhost:8000/PC"));

            foreach (ServiceEndpoint se in host.Description.Endpoints)
                Console.WriteLine("A (address): {0} \nB (binding): {1}
                     \nC (Contract): {2}\n",
                    se.Address, se.Binding.Name, se.Contract.Name);

            host.Open();
            Console.WriteLine("Server in executie. Se asteapta conexiuni...");
            Console.WriteLine("Apasati Enter pentru a opri serverul!");
            Console.ReadKey();
            host.Close();
        }
    }
}
```

Fisier de configurare server App.config (Daca nu exista il adaugam si apoi il completam cu acest cod. Atentie la connectionString caci e diferit pe calculatoarele voastre.)

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configuration, visit
http://go.microsoft.com/fwlink/?LinkID=237468 -->
  </configSections>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.6.1"/>
  </startup>

  <connectionStrings>
    <add name="ModelPostCommentContainer"
connectionString="metadata=res://*/ModelPostComment.csdl|res://*/ModelPostComment.ssdl|re
s://*/ModelPostComment.msl;provider=System.Data.SqlClient;provider connection
string=&quot;data source=IASIMIN-VOSTRO\SQL2012NEW;initial catalog=PostComment;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework&quot;"
providerName="System.Data.EntityClient" />
  </connectionStrings>
  <system.serviceModel>
    <services>
      <service name="ObjectWCF.PostComment" behaviorConfiguration="metadataSupport">
        <!--

        <endpoint address="http://localhost:8000/PC"
                  binding="basicHttpBinding"
                  contract="ObjectWCF.IPostComment"
                  name="BasicHttpBinding_IPostComment">
          <identity>
            <dns value="localhost"/>
          </identity>
        </endpoint>
        <endpoint           address="mex"
                            binding="mexHttpBinding"
                            contract="IMetadataExchange"
                            name="mexhttp"/>
      </service>
    </services>
    <behaviors>
      <serviceBehaviors>
        <behavior name="metadataSupport">
          <!-- Enables the IMetadataExchange endpoint in services that -->
          <!-- use "metadataSupport" in their behaviorConfiguration -->
          <!-- attribute. -->
          <!-- In addition, the httpGetEnabled and httpGetUrl -->
          <!-- attributes publish-->
          <!-- Service metadata for retrieval by HTTP/GET at the address -->
          <!-- "http://192.168.0.102:8000/SampleService?wsdl" -->
          <serviceMetadata httpGetEnabled="true" httpGetUrl=""/>
          <!-- <serviceMetadata/>-->
          <serviceDebug includeExceptionDetailInFaults="true"/>
        </behavior>
      </serviceBehaviors>
    </behaviors>

  </system.serviceModel>
</configuration>
```

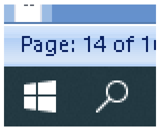
In cadrul aceleasi solutii cream urmatorul proiect pentru client. Clientul va folosi serviciile din HostWCF.

Client. Un nou proiect Windows Forms.

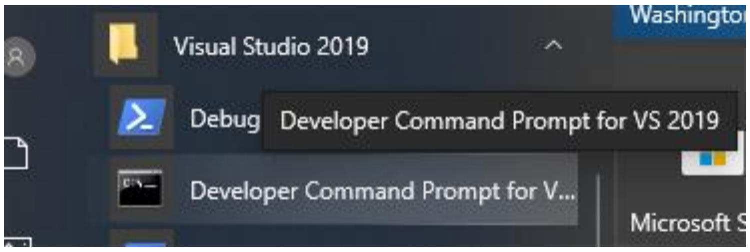
Pregatiri pentru a putea folosi utilitarul svcutil.exe. Cu acest utilitar vom extrage din server metadata necesara pentru a construi clientul.

Cum gasim utilitarul svcutil.exe?

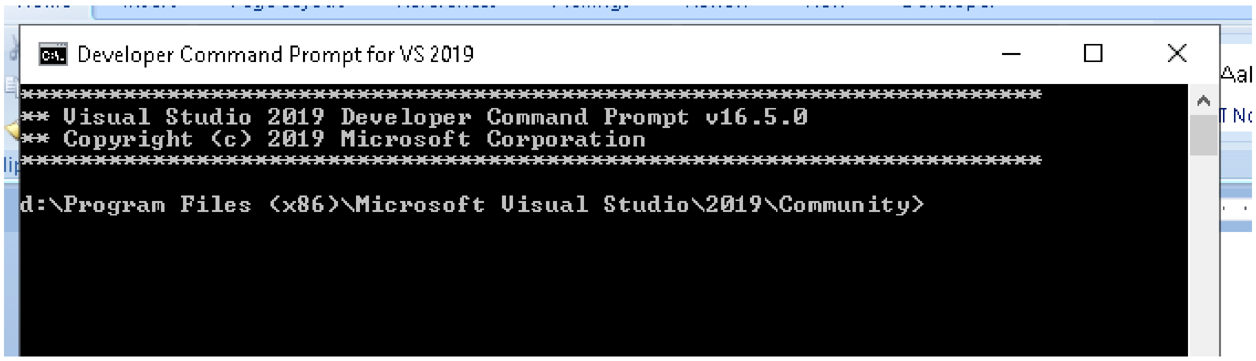
Folosim meniul “Start” (colt stanga jos ecran).



Cautam folderul unde e instalat Visual Studio 2019 si il expandam. Selectam “Developer Command Prompt fro Visual Studio 2019”


Visual Studio 2019
Debug
Developer Command Prompt for VS 2019
Developer Command Prompt for V...
Washingto
Microsoft S

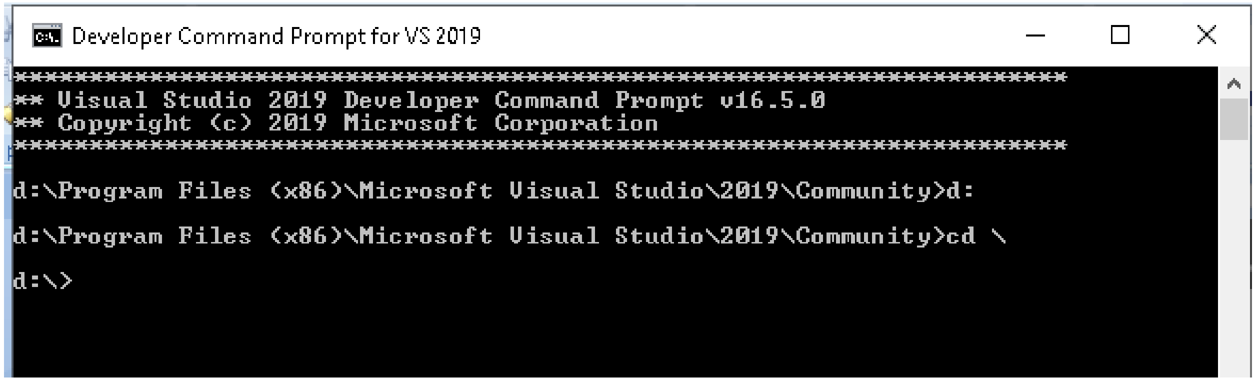
si rezulta


Developer Command Prompt for VS 2019

```
**********************************************************************
** Visual Studio 2019 Developer Command Prompt v16.5.0
** Copyright (c) 2019 Microsoft Corporation
**********************************************************************

d:\Program Files (x86)\Microsoft Visual Studio\2019\Community>
```

Schimbam partitia (aici nu avem drept de scriere). Eu am schimbat pe partitia D:


Developer Command Prompt for VS 2019

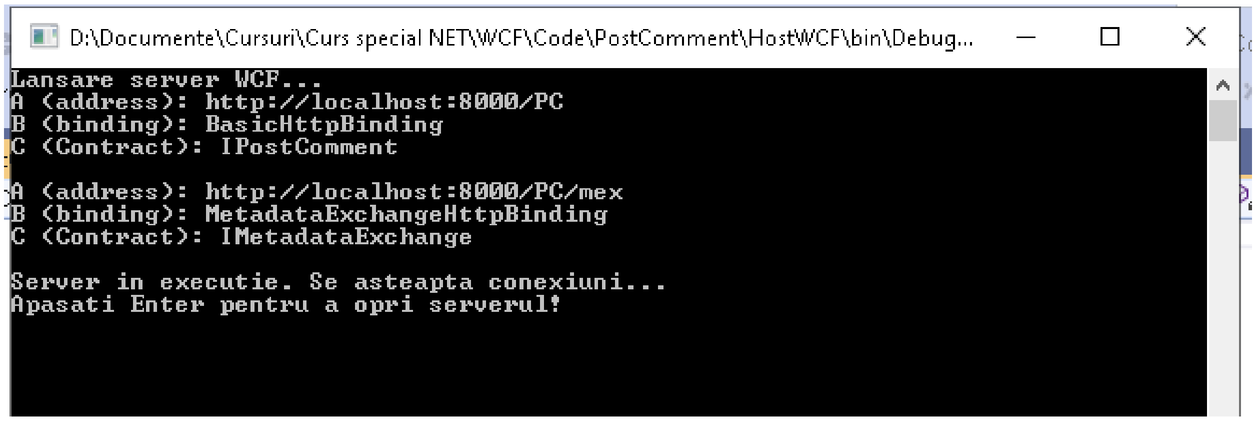
```
**********************************************************************
** Visual Studio 2019 Developer Command Prompt v16.5.0
** Copyright (c) 2019 Microsoft Corporation
**********************************************************************

d:\Program Files (x86)\Microsoft Visual Studio\2019\Community>d:

d:\Program Files (x86)\Microsoft Visual Studio\2019\Community>cd \

d:\>
```

1. Lanasam in executie host-ul, adica codul din etapa a treia a.
2. Va aparea pe ecran ceva de genul:


D:\Documente\Cursuri\Curs special NET\WCF\Code\PostComment\HostWCF\bin\Debug...

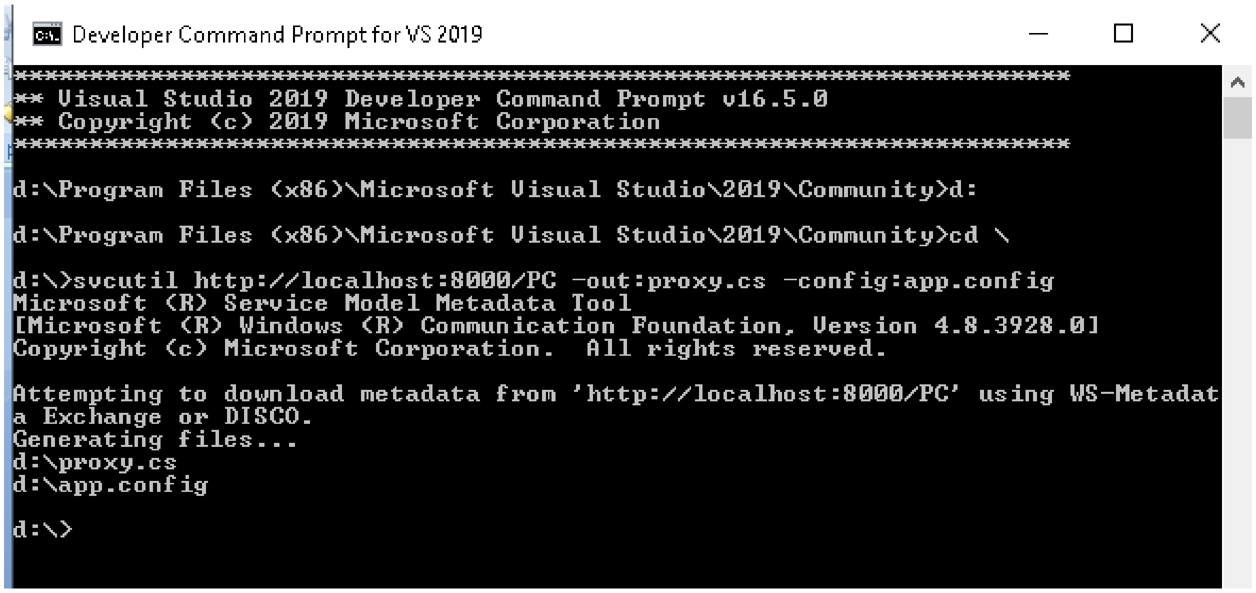
```
Lansare server WCF...
A (address): http://localhost:8000/PC
B (binding): BasicHttpBinding
C (Contract): IPostComment

A (address): http://localhost:8000/PC/mex
B (binding): MetadataExchangeHttpBinding
C (Contract): IMetadataExchange

Server in executie. Se asteapta conexiuni...
Apasati Enter pentru a opri serverul!
```

Obtinere metadata cu svcutil.

In fereastra de comanda deschisa mai inainte tastam urmatoarea comanda:



```
**********************************************************************
** Visual Studio 2019 Developer Command Prompt v16.5.0
** Copyright (c) 2019 Microsoft Corporation
**********************************************************************

d:\Program Files (x86)\Microsoft Visual Studio\2019\Community>d:

d:\Program Files (x86)\Microsoft Visual Studio\2019\Community>cd \

d:\>svcutil http://localhost:8000/PC -out:proxy.cs -config:app.config
Microsoft (R) Service Model Metadata Tool
[Microsoft (R) Windows (R) Communication Foundation, Version 4.8.3928.0]
Copyright (c) Microsoft Corporation.  All rights reserved.

Attempting to download metadata from 'http://localhost:8000/PC' using WS-Metadat
a Exchange or DISCO.
Generating files...
d:\proxy.cs
d:\app.config

d:\>
```

Comanda lansata este:

svcutil http://localhost:8000/PC -out:proxy.cs –config:app.config

Se genereaza doua fisiere (in radacina partitiei D:) proxy.cs si app.config.

Aceste fisiere le vom adauga la solutia pe care o cream in continuare.

Adaugare fisiere generate la proiect.

Deschidem proxy.cs in VS 2019 si vom gasi o clasa PostCommentClient. Instanta acestei clase o vom folosi pentru a apela metodele (operatiile) din server. Vedeti mai jos metoda LoadPosts().

Intr-o app Windows Forms codul arata astfel:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

using PostComment;

namespace ClientPostComment
{
    public partial class Form1 : Form
    {
        List<Post> posts = new List<Post>();

        public Form1()
        {
            InitializeComponent();
        }

       // Handler pentru evenimentul Load al ferestrei principale
        private void Form1_Load(object sender, EventArgs e)
        {
            posts = LoadPosts().ToList<Post>();
            dgp.DataSource = posts;
            dgp.Columns[0].Width = 0;
            if (dgp.Rows.Count > 0)
                dgc.DataSource = posts[0].Comments;
        }

        private static PostComment.Post[] LoadPosts()
        {
            PostCommentClient pc = new PostCommentClient();
            PostComment.Post[] p = pc.GetPosts();
            return p;
        }

       // Handler pentru evenimentul CellMouseClick din DatagridView numit dgp
        private void dgp_CellMouseClick(object sender, DataGridViewCellMouseEventArgs e)
        {
            if (e.RowIndex < 0)
                return;
              // Se afiseaza Comment-urile pentru Post-ul selectat
            dgc.DataSource = null;
            dgc.DataSource = posts[e.RowIndex].Comments;
        }
    }
}
```

Formul principal contine doua DataGridView (dgp pentru Post si dgc pentru Comment). In rest descifrati codul.

Si am terminat.

Cititi si ce urmeaza pentru ...

Observatie: DTO

In client e o problema rezolvata aparent incorect.

Observati ca acest client are referinta la *PostComment*.

Namespace-ul PostComment este luat din proxy.cs si ca atare nu da acces la context.

Continuati ca e interesant...

Ar fi trebuit ca serviciul sa foloseasca DTO (Data Transfer Object) si clientul sa aiba acces la aceste obiecte DTO si nu la cele reale din server.

Rezolvarea ar fi urmatoarea (nu e singura metoda).

Crearea unui contract de date cu DTO. Deci suntem la partea de difinire serviciu.

```
    [DataContract(IsReference = true)]
    public partial class CommentDTO
    {
        [DataMember]
        public int CommentId { get; set; }
        [DataMember]
        public string CommentText { get; set; }
        [DataMember]
        public int PostId { get; set; }
        [DataMember]
        public int PostPostId { get; set; }
        [DataMember]
        public virtual PostDTO Post { get; set; }
    }

    [DataContract]
    public partial class PostDTO
    {
        public PostDTO()
        {
            this.Comments = new List<CommentDTO>(); //new HashSet<Comment>();
        }
        [DataMember]
        public int PostId { get; set; }
        [DataMember]
        public string Title { get; set; }
        [DataMember]
        public virtual List<CommentDTO> Comments { get; set; }
    }
```

```
Definire interfete pentru contractul de servicii. Lucreaza cu DTO.

    [ServiceContract]
    public interface IPost
    {
        [OperationContract]
        void Cleanup();
        [OperationContract]
        PostDTO GetPostById(int id);
        [OperationContract]
        PostDTO GetPostByTitle(string title);
        // Insert, Update, Delete Post
        [OperationContract]
        PostDTO SubmitPost(PostDTO post);
        [OperationContract]
        PostDTO UpdatePost(PostDTO newPost);
        [OperationContract]
        bool DeletePost(int postId);
        [OperationContract]
        List<PostDTO> GetAllPosts();
    }
    [ServiceContract]
    public interface IComment
    {
        // Insert, Update, Delete Comment
        [OperationContract]
        CommentDTO GetCommentById(int id);
        [OperationContract]
        CommentDTO SubmitComment(CommentDTO comment);
        [OperationContract(Name = "AddCommment")]
        CommentDTO SubmitComment(int postId, CommentDTO comment);
        [OperationContract]
        CommentDTO UpdateComment(CommentDTO oldComment, CommentDTO newComment);
        [OperationContract]
        bool DeleteComment(int commentId);
    }

    [ServiceContract]
    public interface ILoadData
    {
        [OperationContract]
        List<PostDTO> GetAllPostsAndRelatedComments();
    }

    [ServiceContract]
    public interface IPostComment : IPost, IComment, ILoadData
    { }
```

Observati ca se lucreaza cu DTO.

Implementare contract de servicii.

Observati aici referinta la EF (using Wcf) si AutoMapper.

Folosesc ***AutoMapper*** pentru DTO.

```
using System;
using System.Collections.Generic;
using System.ServiceModel;
using System.Text;
using Wcf;
using AutoMapper;
using AutoMapper.Configuration;

namespace PostCommentService
{
    [ServiceBehavior(InstanceContextMode = InstanceContextMode.Single)]
    public class ServicePostComment : IPostComment
    {
        /// <summary>
        /// Instanta a clasei ServicePost, clasa ce expune metode de lucru cu EF pentru
        /// baza de date WCF.
        /// Exemplu de cod din aceasta clasa, ServicePost. Vedeti acces la context.
        /// public List<Post> GetAll()
        /// {
```

```
///     List<Post> lp = new List<Post>();
///     using (var context = new ModelPostBlogContainer())
///     {
///         lp = context.Posts.Include(p => p.Comments).ToList();
///     }
///     return lp;
/// }
/// </summary>

private ServicePost _svcPost;

//
// In ctor configurez AutoMapper
//

MapperConfiguration config;
IMapper iMapper;

public ServicePostComment()
{
    _svcPost = new ServicePost();

    // Configurare AutoMapper
    config = new MapperConfiguration(
    cfg => {
        cfg.CreateMap<Post, PostDTO>();
        cfg.CreateMap<Comment, CommentDTO>();
    });
    iMapper = config.CreateMapper();
}

public void Cleanup()
{
    Cleanup();
}

// Interfata ILoadData
public List<PostDTO> GetAllPosts()
{
    var lp = _svcPost.GetAll();
    // Constructie lista PostDTO
    List<PostDTO> lpDto = new List<PostDTO>();
    lpDto = iMapper.Map<List<Post>, List<PostDTO>>(lp);
    return lpDto;
    #region Comentariu
    /* Cam acest lucru face AutoMapper.
      Construieste DTO din obiectele din server (EF)
    foreach (var p in lp)
    {
        PostDTO pd = new PostDTO()
        {
            PostId = p.PostId,
            Title = p.Title
        };
        if (p.Comments.Count > 0)
        {
            foreach (var c in p.Comments)
            {
                //pd.Comments = new List<CommentDTO>();
                CommentDTO cd = new CommentDTO();
                cd.CommentId = c.CommentId;
                cd.CommentText = c.CommentText;
                //cd.PostId = c.PostId;
                //cd.PostPostId = c.PostPostId;
                pd.Comments.Add(cd);
            }
        }
        lpDto.Add(pd);
    }
    return lpDto;
    */
    #endregion
}


public void DeleteComment(CommentDTO comment)
{
    Comment comm = new Comment();
    // map CommentDTO la Comment
    comm = iMapper.Map<CommentDTO, Comment>(comment);
    _svcPost.DeleteComment(comm);
}

public PostDTO GetPostByTitle(string title)
{
    Post post = _svcPost.GetPostByTitle(title);
    if (post != null)
    {
        PostDTO postDTO = iMapper.Map<Post, PostDTO>(post);
        return postDTO;
    }
    return null;
}

// IPost implementation methods

public PostDTO GetPostById(int id)
{
    Wcf.Post post = _svcPost.GetPostById(id);
    // Reconstructie obiecte cunoscute in serviciu
    PostDTO postDTO = iMapper.Map<Post, PostDTO>(post);
    return postDTO;
}

public PostDTO SubmitPost(PostDTO postDTO)
{
    Post post = new Post();
    post = iMapper.Map<PostDTO, Post>(postDTO);
    post = _svcPost.SubmitPost(post);
    postDTO = iMapper.Map<Post, PostDTO>(post);
    return postDTO;
}

public PostDTO UpdatePost(PostDTO newPost)
{
    Post post = iMapper.Map<PostDTO, Post>(newPost);
```

```
            post = _svcPost.UpdatePost(post);
            PostDTO postDTO = iMapper.Map<Post, PostDTO>(post);
            return postDTO;
        }

        public bool DeletePost(int postId)
        {
            return _svcPost.DeletePost(postId);
        }

        // IComment implementation method
        public CommentDTO GetCommentById(int id)
        {
            Comment comment = _svcPost.GetCommentById(id);
            CommentDTO commentDTO = iMapper.Map<Comment, CommentDTO>(comment);
            return commentDTO;
        }

        public CommentDTO SubmitComment(CommentDTO commentDTO)
        {
            Comment comment = iMapper.Map<CommentDTO, Comment>(commentDTO);
            comment = _svcPost.SubmitComment(comment);
            CommentDTO commDTO = iMapper.Map<Comment, CommentDTO>(comment);
            return commDTO;
        }

        public CommentDTO SubmitComment(int postId, CommentDTO commentDTO)
        {
            Comment comment = iMapper.Map<CommentDTO, Comment>(commentDTO);
            comment = _svcPost.SubmitComment(postId, comment);
            CommentDTO comm = iMapper.Map<Comment, CommentDTO>(comment);
            return comm;
        }

        public CommentDTO UpdateComment(CommentDTO oldCommentDTO,
              CommentDTO newCommentDTO)
        {
            Comment oldComment = iMapper.Map<CommentDTO, Comment>(oldCommentDTO);
            Comment newComment = iMapper.Map<CommentDTO, Comment>(newCommentDTO);
            Comment comment = _svcPost.UpdateComment(oldComment, newComment);
            CommentDTO comm = iMapper.Map<Comment, CommentDTO>(comment);
            return comm;
        }

        public bool DeleteComment(int commentId)
        {
            return _svcPost.DeleteComment(commentId);
        }

        List<PostDTO> ILoadData.GetAllPostsAndRelatedComments()
        {
            // redirectare !!!!????
            return GetAllPosts();
        }
    }

}
```

INTREBARI?????