Lucrare de laborator nr. 6
Arhitecturi orientate pe servicii

6.1 Scopul lucrării

În cadrul acestei lucrări de laborator se urmărește consolidarea abilităților în utilizarea eficientă a șabloanelor arhitecturale orientate pe servicii.

6.2 Exerciții rezolvate

Exercițiu 6.1. Dezvoltați, utilizând o arhitectură orientată pe servicii, o aplicație soft care să permită gestionarea listei angajaților dintr-o firmă.

Soluție:

Soluția prezentată utilizează o arhitectură bazată pe servicii WCF (figura 6.1).

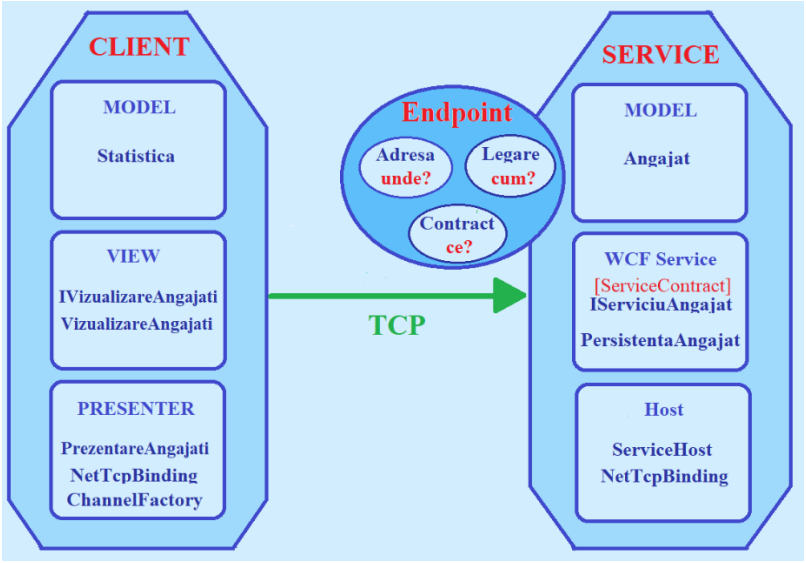

Figura 6.1. Arhitectura utilizând servicii WCF

Diagrama de clase a aplicației server are structura prezentată în figura 6.2. Un serviciu este o clasă care expune funcționalitățile disponibile clienților la unul sau mai multe puncte finale. În acest scop, pentru a crea un serviciu asociat clasei ***Angajat***, s-a implementat clasa ***PersistentaAngajat*** care implementează contractul definit de interfața ***IServiciuAngajat***.

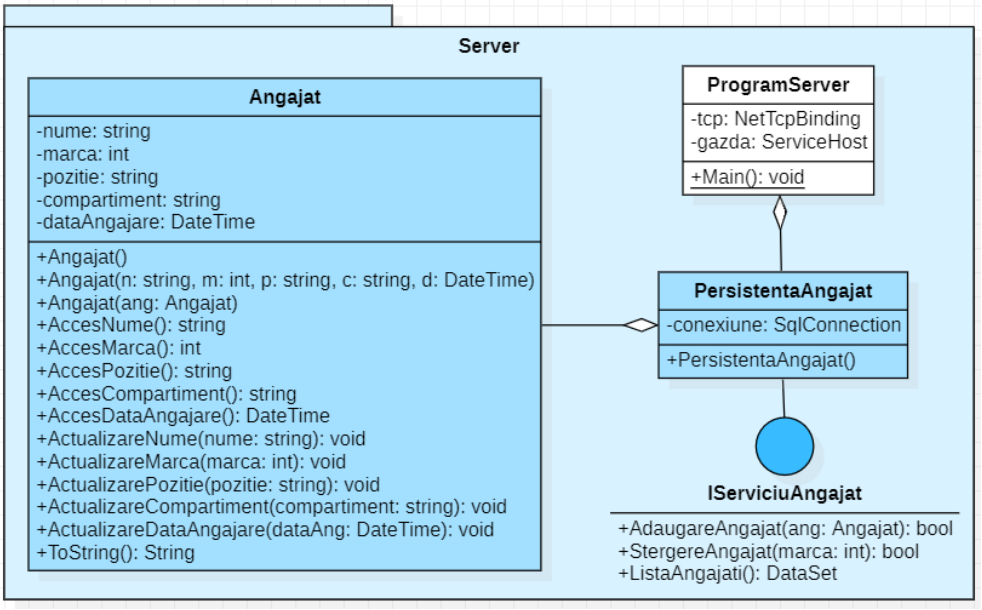

Figura 6.2. Diagramă de clase

Implementarea interfeței ***IServiciuAngajat*** este următoarea:

```
using System.ServiceModel;
using System.Data;
namespace Server
{
    // Definirea contractului IServiciuAngajat
    [ServiceContract]
    public interface IServiciuAngajat
    {
        [OperationContract]
        bool AdaugareAngajat(Angajat ang);
        [OperationContract]
        bool StergereAngajat(int marca);
        [OperationContract]
        DataSet ListaAngajati();
    }
}
```

Implementarea clasei ***PersistentaAngajat*** este următoarea:

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.ServiceModel;
using System.Configuration;
namespace Server
{
```

```
    [ServiceBehavior(ConcurrencyMode = ConcurrencyMode.Multiple,
UseSynchronizationContext = false, InstanceContextMode =
InstanceContextMode.Single)]
    // Implementarea contractului IServiciuAngajat
    internal class PersistentaAngajat : IServiciuAngajat
    {
        SqlConnection conexiune;
        public PersistentaAngajat()
        {
            conexiune = null;
        }
        public bool AdaugareAngajat(Angajat ang)
        {
            string dataAngajare = ang.DataAngajare.Year.ToString() +
"-" + ang.DataAngajare.Month.ToString() + "-" +
ang.DataAngajare.Day.ToString();
            bool rezultat = true;
            try
            {
                string s = ConfigurationManager.ConnectionStrings
["Angajati"].ConnectionString;
                conexiune = new SqlConnection(s);
                if (conexiune.State == ConnectionState.Closed)
                    conexiune.Open();
                SqlCommand adaugare = new SqlCommand("insert into
Angajat values('" + ang.Nume + "','" + ang.Marca + "','" +
ang.Pozitie + "','" + ang.Compartiment + "','" + dataAngajare +
"')", conexiune);
                if (adaugare.ExecuteNonQuery() == 0)
                    rezultat = false;
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
                rezultat = false;
            }
            finally
            {
                conexiune.Close();
            }
            return rezultat;
        }
        public bool StergereAngajat(int marca)
        {
            bool rezultat = true;
            try
            {
                string s = ConfigurationManager.ConnectionStrings
["Angajati"].ConnectionString;
                conexiune = new SqlConnection(s);
                if (conexiune.State == ConnectionState.Closed)
                    conexiune.Open();
                SqlCommand stergere = new SqlCommand("delete from
Angajat where Marca = '" + marca + "'", conexiune);
                if (stergere.ExecuteNonQuery() == 0)
```

```
                    rezultat = false;
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
                rezultat = false;
            }
            finally
            {
                conexiune.Close();
            }
            return rezultat;
        }
        public DataSet ListaAngajati()
        {
            try
            {
                string s = ConfigurationManager.ConnectionStrings
["Angajati"].ConnectionString;
                conexiune = new SqlConnection(s);
                if (conexiune.State == ConnectionState.Closed)
                    conexiune.Open();
                SqlCommand vizualizare = new SqlCommand("Select *
from Angajat sort order by Nume", conexiune);
                SqlDataAdapter dateCitite = new
SqlDataAdapter(vizualizare);
                DataSet ds = new DataSet();
                dateCitite.Fill(ds, "vizualizareAng");
                return ds;
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
                return null;
            }
            finally
            {
                conexiune.Close();
            }
        }
    }
}
```

După implementarea contractului de servicii ***IServiciuAngajat***, trebuie implementată o clasă (***ProgramServer***) care permite crearea de puncte finale pentru serviciu. Fiecare punct final conține o adresă care indică unde să găsească punctul final, o legare care specifică modul în care un client poate comunica cu punctul final și un contract care identifică metodele disponibile. Legarea specifică modul în care punctul final comunică cu clienții, inclusiv ce protocol de transport să folosească (ex: TCP sau HTTP) și ce codificare să folosească pentru mesaje (ex: text sau binar).

În această soluție se utilizează pentru legare o instanță a clasei ***NetTcpBinding***. Aceasta generează o stivă de comunicații în timpul rulării în mod implicit care utilizează securitatea transportului ***TCP*** pentru livrarea mesajelor și o codificare binară a mesajului.

Implementarea clasei ***ProgramServer*** este următoarea:

```
using System;
using System.ServiceModel;
using System.Configuration;
namespace Server
{
    class ProgramServer
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Conectare la baza de date ...");
            NetTcpBinding tcp = new NetTcpBinding();
            tcp.OpenTimeout = new TimeSpan(0, 60, 0);
            tcp.SendTimeout = new TimeSpan(0, 60, 0);
            tcp.ReceiveTimeout = new TimeSpan(0, 60, 0);
            tcp.CloseTimeout = new TimeSpan(0, 60, 0);
            tcp.MaxReceivedMessageSize = System.Int32.MaxValue;
            tcp.ReaderQuotas.MaxArrayLength = System.Int32.MaxValue;
            string s = ConfigurationManager.ConnectionStrings
["AdresaIP"].ConnectionString;
            tcp.Security.Transport.ClientCredentialType =
TcpClientCredentialType.Windows;
            tcp.Security.Message.ClientCredentialType =
MessageCredentialType.Certificate;
            PersistentaAngajat ang = new PersistentaAngajat();
 // Instantierea unui obiect ServiceHost pentru serviciul PersistentaAngajat
            ServiceHost gazda = new ServiceHost(ang);
            try
            {
                gazda.AddServiceEndpoint(typeof(IServiciuAngajat),
tcp, "net.tcp://" + s + ":52001/Angajati");
                gazda.Open();
                Console.WriteLine("Conectare realizata.");
                Console.ReadLine();
            }
            catch (Exception ex)
            {
                Console.WriteLine("Nu s-a realizat conectarea la
baza de date. " + ex.ToString());
                Console.ReadLine();
            }
            finally
            {
                gazda.Close();
            }
        }
    }
}
```

Diagrama de clase a aplicației client are structura prezentată în figura 6.3. Aceasta conține o interfață (***IVizualizareAngajati***) și 4 clase: ***VizualizareAngajati***, ***PrezentareAngajati***, ***Statistica*** și ***ProgramClient***.

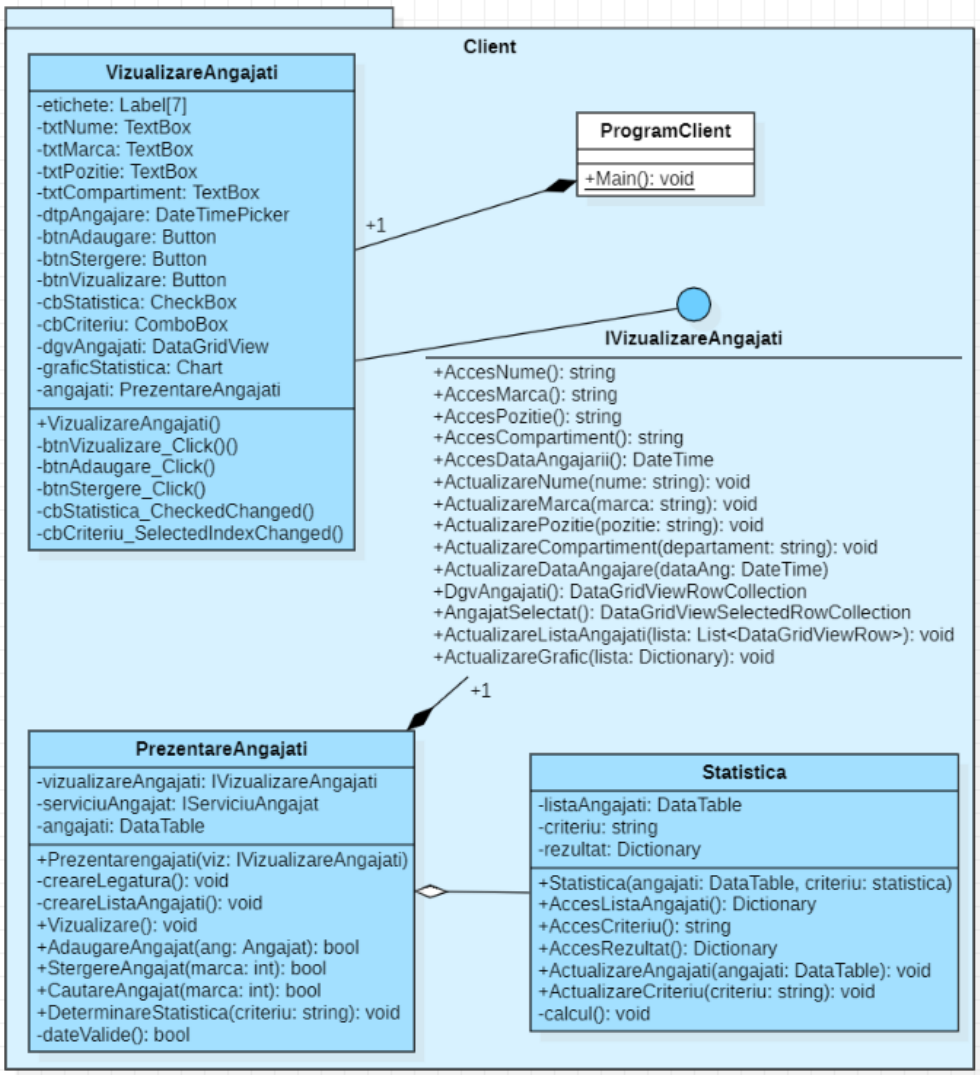

Figura 6.3. Diagramă de clase

Aplicația client WCF (Windows Communication Foundation) folosește clasa ***ChannelFactory<IServiciuAngajat>*** pentru a crea un canal de comunicare cu serviciu WCF implementat de aplicația server. Implementarea clasei ***PrezentareAngajati*** este următoarea:

```
using Server;
using System;
using System.Data;
using System.ServiceModel;
using System.Configuration;
using System.Windows.Forms;
using System.Collections.Generic;
namespace Client
{
    public class PrezentareAngajati
    {
        private IVizualizareAngajati vizualizareAng;
        private IServiciuAngajat serviciuAngajat;
        private DataTable angajati;
        public PrezentareAngajati(IVizualizareAngajati vizAng)
        {
            this.vizualizareAng = vizualizareAng;
            this.angajati = new DataTable();
            this.creareLegatura();
        }
        private void creareLegatura()
        {
            ChannelFactory<IServiciuAngajat> canalAngajat;
            NetTcpBinding tcp = new NetTcpBinding();
            tcp.OpenTimeout = new TimeSpan(0, 60, 0);
            tcp.SendTimeout = new TimeSpan(0, 60, 0);
            tcp.ReceiveTimeout = new TimeSpan(0, 60, 0);
            tcp.CloseTimeout = new TimeSpan(0, 60, 0);
            tcp.MaxReceivedMessageSize = System.Int32.MaxValue;
            tcp.Security.Mode = SecurityMode.Transport;
            tcp.Security.Transport.ClientCredentialType =
TcpClientCredentialType.Windows;
            tcp.Security.Transport.ProtectionLevel =
System.Net.Security.ProtectionLevel.EncryptAndSign;
            string s = ConfigurationManager.ConnectionStrings
["AdresaIP"].ConnectionString;
            canalAngajat = new ChannelFactory<IServiciuAngajat>(tcp,
"net.tcp://" + s + ":52001/Angajati");
            try
            {
                this.serviciuAngajat = canalAngajat.CreateChannel();
                this.creareListaAngajati();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());
            }
        }
        private void creareListaAngajati()
        {
            try
            {
                DataSet ds = this.serviciuAngajat.ListaAngajati();
                this.angajati = ds.Tables["vizualizareAng"];
            }
```

```
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());
            }
        }
        public void Vizualizare()
        {  ...  }
        public bool CautareAngajat(int marca)
        {  ...  }
        private bool dateValide()
        {  ...  }
        public void AdaugareAngajat()
        {  ...  }
        public void StergereAngajat()
        {  ...  }
        public void DeterminareStatistica(string criteriu)
        {  ...  }
    }
}
```

Interfața grafică a aplicației client (instanță a clasei ***VizualizareAngajati***) este prezentată în figura 6.4.

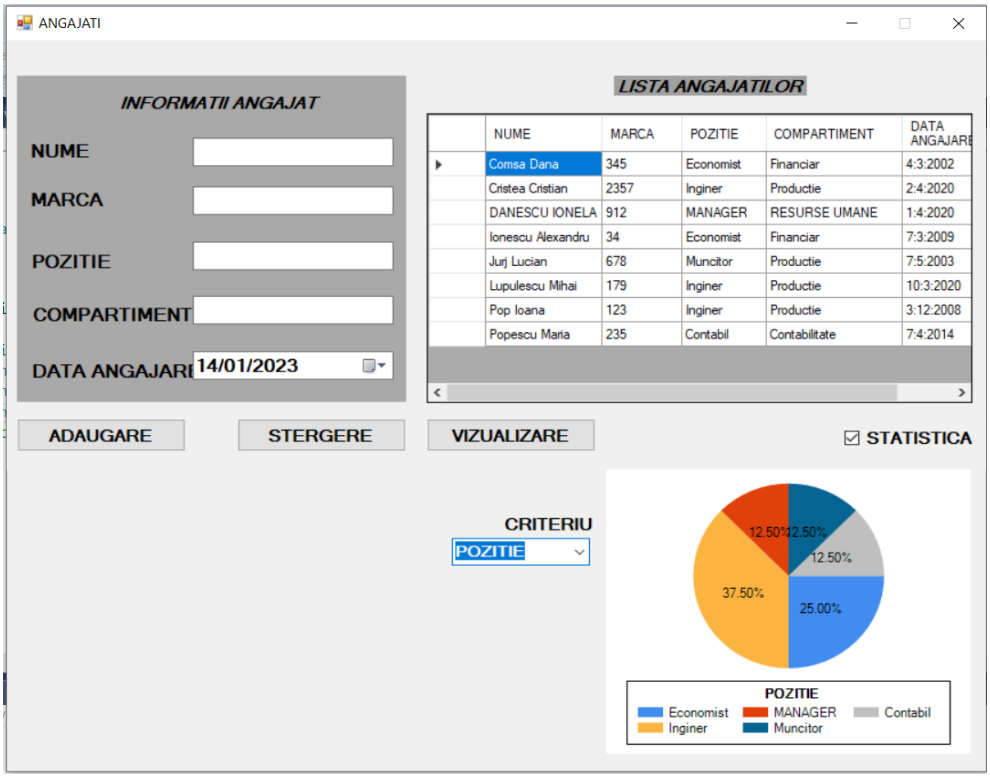

Figura 6.4. Interfața grafică

6.3 Exerciții propuse spre rezolvare

Exercițiu 6.2. Dezvoltați, utilizând o arhitectură orientată pe servicii, un sistem software pentru managementul operațiilor dintr-o tipografie. Tipografia furnizează servicii de comercializare a hârtiei: tipărire, tăiere, legare şi lipire. Aceste servicii pot fi combinate rezultând servicii mai complexe. Fiecare serviciu are un anumit preţ. La acest preţ se adaugă adaosul comercial al tipografiei, dar se poate aplica şi un discount pentru comenzi mari. Sistemul soft trebuie să permită:

- Preluarea informațiilor despre client;
- Administrarea serviciilor simple (specificare denumire, preţ, adaos comercial, discount aplicat);
- Administrarea serviciilor compuse (servicii componente, discount aplicat);
- Generare rapoarte:
  - Grafic comenzi livrate/anulate;
  - Grafic servicii simple/servicii compuse.