

# PROGRAMACIÓN III

---

Tarea 2 - 11/09/2017

Docente: Adriana Cabella

Estudiantes: Guillermo Polachek (153924) – Sebastián Villar (177751)

## Contenido

Clase Proveedor .....	2
Clase Servicio .....	8
Clase TipoEvento .....	12
WCF Exponer Catálogo de Servicios.....	13
WCF Agregar Proveedor .....	15
WCF Lista Proveedores .....	17

## Clase Proveedor

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
using System.Diagnostics;

namespace Dominio
{
    public class Proveedor : IActiveRecord
    {
        #region Propiedades
        public string RUT { get; set; }
        public string NombreFantasia { get; set; }
        public string Email { get; set; }
        public Usuario MiUsuario { get; set; } = new Usuario();
        public string Telefono { get; set; }
        public string FechaRegistro { get; set; }
        public bool esInactivo { get; set; }
        public static double Arancel { get; set; }
        public double Arancel11 { get; set; } // Solo para test con BD
        //public int porcentajeExtra { get; set; }
        public bool esVip { get; set; }
        #endregion

        public override string ToString()
        {
            string ret = string.Format("{0} {1}", "Rut: " + RUT + " - ",
                "NombreFantasia:" + NombreFantasia);
            return ret;
        }

        #region Métodos de lógica
        public virtual bool Validar()
        {
            return this.RUT.Length == 12 && this.NombreFantasia.Length > 3
                && this.Email.Length > 3 && this.Telefono.Length > 3;
        }
        #endregion
    }
}
```

```

#region Manejo de Usuario
public bool AgregarUsuario(Usuario usu)
{
    this.MiUsuario = usu;
    return true;
}
#endregion

#region Acceso a datos
public bool Insertar()
{
    SqlConnection cn = null;
    if (!this.Validar()) return false;
    SqlTransaction trn = null;

    cn = Conexion.CrearConexion();
    cn.Open();
    trn = cn.BeginTransaction();

    try
    {
        SqlCommand cmd = new SqlCommand(
            @"INSERT INTO Proveedor
            VALUES (@rut, @nombrefantasia, @email, @telefono,
            @arancel, @fecharegistro, @esInactivo, @esVip);
            SELECT CAST (SCOPE_IDENTITY() AS INT)", cn);
        cmd.Parameters.AddWithValue("@RUT", this.RUT);
        cmd.Parameters.AddWithValue("@nombreFantasia",
            this.NombreFantasia);
        cmd.Parameters.AddWithValue("@email", this.Email);
        cmd.Parameters.AddWithValue("@telefono", this.Telefono);
        cmd.Parameters.AddWithValue("@arancel", this.Arancel);
        cmd.Parameters.AddWithValue("@fechaRegistro",
            this.FechaRegistro);
        cmd.Parameters.AddWithValue("@esInactivo",
            this.esInactivo);
        cmd.Parameters.AddWithValue("@esVip", this.esVip);

        cmd.Transaction = trn;
        cmd.ExecuteNonQuery();

        cmd.CommandText = @"INSERT INTO Usuario
            VALUES(@usuario,@password,@rol)";
        cmd.Parameters.Clear();
        cmd.Parameters.AddWithValue("@usuario", MiUsuario.User);
        cmd.Parameters.AddWithValue("@password", MiUsuario.Passw);
        cmd.Parameters.AddWithValue("@rol", 2);
        cmd.ExecuteNonQuery();

        if (esVip)
        {

```

```

        cmd.CommandText = @"INSERT INTO ProveedorVip
                                VALUES(@idProveedor,@porcentajeExtra)";
        cmd.Parameters.Clear();
        cmd.Parameters.AddWithValue("@idProveedor", this.RUT);
        cmd.Parameters.AddWithValue("@porcentajeExtra", 5);
        cmd.ExecuteNonQuery();
    }

    trn.Commit();
    return true;
}
catch (Exception ex)
{
    System.Diagnostics.Debug.Assert(false, "Error: " + ex.Message);
    return false;
}
finally { cn.Close(); cn.Dispose(); trn.Dispose(); }
}

public bool Eliminar()
{
    string cadenaDelete = @"DELETE Proveedor WHERE RUT=@rut;";
    SqlCommand cmd = new SqlCommand();
    cmd.CommandText = cadenaDelete;
    cmd.Parameters.Add(new SqlParameter("@rut", this.RUT));
    SqlConnection cn = Conexion.CrearConexion();
    try
    {
        Conexion.AbrirConexion(cn);
        cmd.ExecuteNonQuery();
        return true;
    }
    catch (Exception ex)
    {
        Debug.Assert(false, ex.Message);
        return false;
    }
    finally
    {
        Conexion.CerrarConexion(cn);
    }
}

public bool Modificar()
{
    throw new NotImplementedException();
}
#endregion

#region Finders
public static Proveedor FindByRUT(string rut)
{
    SqlConnection cn = Conexion.CrearConexion();
    SqlCommand cmd = new SqlCommand(@"SELECT * From Proveedor
                                    WHERE Rut = @rut");

```

```

cmd.Connection = cn;
cmd.Parameters.AddWithValue("@rut", rut);
try
{
    cn.Open();
    SqlDataReader dr = cmd.ExecuteReader();
    if (dr.HasRows)
    {
        if (dr.Read())
        {
            Proveedor p = new Proveedor
            {
                RUT = rut,
                NombreFantasia =
                    dr["NombreFantasia"].ToString(),
                Email = dr["Email"].ToString(),
            };
            return p;
        }
    }
    return null;
}
catch (Exception ex)
{
    throw new Exception("No existe el Proveedor");
}
finally { cn.Close(); cn.Dispose(); }
}

public static Proveedor FindByEmail(string email)
{
    SqlConnection cn = Conexion.CrearConexion();
    SqlCommand cmd = new SqlCommand(@"SELECT * From Proveedor WHERE Email
= @email");
    cmd.Connection = cn;
    cmd.Parameters.AddWithValue("@email", email);
    try
    {
        cn.Open();
        SqlDataReader dr = cmd.ExecuteReader();
        if (dr.HasRows)
        {
            if (dr.Read())
            {
                Proveedor p = new Proveedor
                {
                    RUT = dr["RUT"].ToString(),
                    NombreFantasia =
                        dr["NombreFantasia"].ToString(),
                    Email = email,
                };
                return p;
            }
        }
    }
}

```

```

        }
        return null;
    }
    catch (Exception ex)
    {
        throw new Exception("No existe el Proveedor");
    }
    finally { cn.Close(); cn.Dispose(); }
}

public static List<Proveedor> FindAll()
{
    SqlConnection cn = Conexion.CrearConexion();
    SqlCommand cmd = new SqlCommand();
    cmd.CommandText = @"SELECT * FROM Proveedor";
    cmd.Connection = cn;
    List<Proveedor> listaProveedores = null;
    try
    {
        Conexion.AbrirConexion(cn);
        SqlDataReader dr = cmd.ExecuteReader();
        if (dr.HasRows)
        {
            listaProveedores = new List<Proveedor>();
            while (dr.Read())
            {
                Proveedor p = CargarDatosDesdeReader(dr);
                listaProveedores.Add(p);
            }
        }
        return listaProveedores;
    }
    catch (SqlException ex)
    {
        System.Diagnostics.Debug.Assert(false, ex.Message);
        return null;
    }
    finally
    {
        Conexion.CerrarConexion(cn);}
}

protected static Proveedor CargarDatosDesdeReader(IDataRecord fila)
{
    Proveedor p = null;
    if (fila != null)
    {
        p = new Proveedor
        {
            RUT = fila.IsDBNull(fila.GetOrdinal("Rut")) ? "" :
fila.GetString(fila.GetOrdinal("Rut")),
            NombreFantasia =

```

```

        fila.IsDBNull(fila.GetOrdinal("NombreFantasia")) ? "" :
        fila.GetString(fila.GetOrdinal("NombreFantasia")),
        Email = fila.IsDBNull(fila.GetOrdinal("Email")) ? "" :
        fila.GetString(fila.GetOrdinal("Email")),
    };
}
return p;
}
#endregion
}
}

```



## Clase Servicio

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
using System.Diagnostics;

namespace Dominio
{
    public class Servicio : IActiveRecord
    {
        public string Nombre { get; set; }
        public string Foto { get; set; }
        public string Descripcion { get; set; }
        public List<TipoEvento> ListaTipoEventos = new List<TipoEvento>();

        public override string ToString()
        {
            string ret = string.Format("{0}", Nombre);
            return ret;
        }

        #region Acceso a datos
        public bool Insertar()
        {
            throw new NotImplementedException();
        }

        public bool Eliminar()
        {
            throw new NotImplementedException();
        }

        public bool Modificar()
        {
            throw new NotImplementedException();
        }
        #endregion

        #region Finders

        public static Servicio FindByNombre(string nombre)
        {
            SqlConnection cn = Conexion.CrearConexion();
```

```

        SqlCommand cmd = new SqlCommand(@"SELECT * From Servicio WHERE Nombre
= @nombre");
        cmd.Connection = cn;
        cmd.Parameters.AddWithValue("@nombre", nombre);
        try
        {
            cn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            if (dr.HasRows)
            {
                if (dr.Read())
                {
                    string nombreServicio =
dr.IsDBNull(dr.GetOrdinal("nombre")) ? "" : dr.GetString(dr.GetOrdinal("nombre"));
                    string desc = dr.IsDBNull(dr.GetOrdinal("Descripcion")) ?
"" : dr.GetString(dr.GetOrdinal("Descripcion"));
                    string foto = dr.IsDBNull(dr.GetOrdinal("imagen")) ? "" :
dr.GetString(dr.GetOrdinal("imagen"));

                    Servicio s = new Servicio
                    {
                        Nombre = nombreServicio,
                        Descripcion = desc,
                        Foto = foto,
                        ListaTipoEventos = new List<TipoEvento>()
                    };
                    return s;
                }
                return null;
            }
        }
        catch (Exception ex)
        {
            throw new Exception("No existe el Servicio");
        }
        finally { cn.Close(); cn.Dispose(); }
    }

    public static List<Servicio> FindAll()
    {
        SqlConnection cn = Conexion.CrearConexion();
        SqlCommand cmd = new SqlCommand();

        cmd.CommandText = @"SELECT s.nombre AS Servicio, s.descripcion AS
'Descripción del servicio', s.imagen as 'Foto', t.nombre as 'Tipo de evento'
FROM Servicio AS s
INNER JOIN TipoEventoYServicio AS e ON
s.idServicio = e.idServicio
INNER JOIN TipoEvento AS t ON e.idTipoEvento =
t.idTipoEvento";
        cmd.Connection = cn;
        List<Servicio> listaServicios = null;
        try

```

```

    {
        Conexion.AbrirConexion(cn);
        SqlDataReader dr = cmd.ExecuteReader();
        if (dr.HasRows)
        {
            listaServicios = new List<Servicio>();
            while (dr.Read())
            {
                Servicio s = CargarDatosDesdeReader(dr);
                listaServicios.Add(s);
            }
        }
        return listaServicios;
    }
    catch (SqlException ex)
    {
        //
        System.Diagnostics.Debug.Assert(false, ex.Message);
        return null;
    }
    finally
    {
        Conexion.CerrarConexion(cn);
    }
}

public static List<TipoEvento> FindTiposEventoByServicio(string servicio)
{
    SqlConnection cn = Conexion.CrearConexion();
    SqlCommand cmd = new SqlCommand();

    cmd.CommandText = @"SELECT t.nombre, t.descripcion
                        FROM Servicio AS s
                        INNER JOIN TipoEventoYServicio AS e ON
s.idServicio = e.idServicio
                        INNER JOIN TipoEvento AS t ON e.idTipoEvento =
t.idTipoEvento
                        WHERE s.nombre = @servicio";

    cmd.Connection = cn;
    cmd.Parameters.AddWithValue("@servicio", servicio);
    List<TipoEvento> listaTipoEvento = null;
    try
    {
        Conexion.AbrirConexion(cn);
        SqlDataReader dr = cmd.ExecuteReader();
        if (dr.HasRows)
        {
            listaTipoEvento = new List<TipoEvento>();
            while (dr.Read())
            {

```

```

        Servicio s = Servicio.FindByNombre(servicio);
        string tipo = dr.IsDBNull(dr.GetOrdinal("nombre")) ? "" :
dr.GetString(dr.GetOrdinal("nombre"));
        string desc = dr.IsDBNull(dr.GetOrdinal("descripción")) ?
"" : dr.GetString(dr.GetOrdinal("descripción"));
        TipoEvento t = new TipoEvento(tipo, desc);
        listaTipoEvento.Add(t);
    }
}
return listaTipoEvento;
}
catch (SqlException ex)
{
    //
    System.Diagnostics.Debug.Assert(false, ex.Message);
    return null;
}
finally
{
    Conexion.CerrarConexion(cn);
}
}

protected static Servicio CargarDatosDesdeReader(IDataRecord fila)
{
    Servicio s = null;
    string nombreServicio = fila.IsDBNull(fila.GetOrdinal("Servicio")) ?
"" : fila.GetString(fila.GetOrdinal("Servicio"));
    string desc = fila.IsDBNull(fila.GetOrdinal("Descripción del
servicio")) ? "" : fila.GetString(fila.GetOrdinal("Descripción del servicio"));
    string foto = fila.IsDBNull(fila.GetOrdinal("Foto")) ? "" :
fila.GetString(fila.GetOrdinal("Foto"));

    if (fila != null)
    {
        s = new Servicio()
        {
            Nombre = nombreServicio,
            Descripcion = desc,
            Foto = foto,
            ListaTipoEventos = FindTiposEventoByServicio(nombreServicio)
        };
    }
    return s;
}
#endregion
}
}

```

## Clase TipoEvento.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
using System.Diagnostics;

namespace Dominio
{
    public class TipoEvento
    {
        private string tipo;
        private string desc;

        public TipoEvento(string tipo, string desc)
        {
            this.tipo = tipo;
            this.desc = desc;
        }

        public string Nombre { get; set; }
        public string Descripcion { get; set; }
    }
}
```

# WCF EXPONER CATÁLOGO DE SERVICIOS

## IServicioCatalogoServicios.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;
using Dominio;

namespace WcfServicioCatalogoServicios
{
    // NOTA: puede usar el comando "Rename" del menú "Refactorizar" para cambiar
    el nombre de interfaz "IService1" en el código y en el archivo de configuración a
    la vez.
    [ServiceContract]
    public interface IServicioCatalogoServicios
    {
        [OperationContract]
        IEnumerable<DtoServicio> ObtenerServicios();
    }

    [DataContract]
    public class DtoServicio
    {
        [DataMember]
        public string Servicio { get; set; }

        [DataMember]
        public string Descripcion { get; set; }

        [DataMember]
        public string Foto { get; set; }

        [DataMember]
        public List<TipoEvento> TipoEvento { get; set; }
    }
}
```

## ServicioCatalogoServicios.svc

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;
using Dominio;

namespace WcfServicioCatalogoServicios
{
    // NOTA: puede usar el comando "Rename" del menú "Refactorizar" para cambiar
    // el nombre de clase "Service1" en el código, en svc y en el archivo de
    // configuración.
    // NOTE: para iniciar el Cliente de prueba WCF para probar este servicio,
    // seleccione Service1.svc o Service1.svc.cs en el Explorador de soluciones e inicie
    // la depuración.
    public class ServicioCatalogoServicios : IServicioCatalogoServicios
    {
        public IEnumerable<DtoServicio> ObtenerServicios()
        {
            List<Servicio> listaCompleta = Servicio.FindAll();
            if (listaCompleta == null) return null;
            List<DtoServicio> servicios = new List<DtoServicio>();
            foreach (Servicio s in listaCompleta)
            {
                List<TipoEvento> listaTipoEvento =
                Servicio.FindTiposEventoByServicio(s.Nombre);
                servicios.Add(
                    new DtoServicio()
                    {
                        Servicio = s.Nombre,
                        Descripcion = s.Descripcion,
                        Foto = s.Foto,
                        TipoEvento = listaTipoEvento
                    }
                );
            }
            return servicios;
        }
    }
}
```

## WCF AGREGAR PROVEEDOR

### IAgregarProv.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;
using Dominio;

namespace WcfAgregarProv
{
    [ServiceContract]
    public interface IAgregarProv
    {
        [OperationContract]
        bool InsertarProveedor(string rut, string nombreFantasia, string email,
string tel, double arancel, string fechaRegistro, bool esInactivo, bool esVip,
string pass);
    }
}
```



## ServicioAgregarProv.svc

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;
using Dominio;

namespace WcfAgregarProv
{
    // NOTA: puede usar el comando "Rename" del menú "Refactorizar" para cambiar
    el nombre de clase "Service1" en el código, en svc y en el archivo de
    configuración.
    // NOTE: para iniciar el Cliente de prueba WCF para probar este servicio,
    seleccione Service1.svc o Service1.svc.cs en el Explorador de soluciones e inicie
    la depuración.
    public class ServicioAgregarProv : IAgregarProv
    {
        public bool InsertarProveedor(string rut, string nombreFantasia, string
        email, string tel, double arancel, string fechaRegistro, bool esInactivo, bool
        esVip, string pass)
        {
            // Construyo un proveedor con los parámetros que llegan desde el
            servicio
            Proveedor p = new Proveedor()
            {
                RUT = rut,
                NombreFantasia = nombreFantasia,
                Email = email,
                Telefono = tel,
                Arancel = arancel,
                FechaRegistro = fechaRegistro,
                esInactivo = esInactivo,
                esVip = esVip
            };

            // Encripto el password y construyo un usuario
            string passEncriptada = Usuario.EncriptarPassSHA512(pass);
            Usuario usu = new Usuario { User = rut, Passw = passEncriptada };

            // Agrego el usuario al proveedor p
            p.AgregarUsuario(usu);

            return p.Insertar();
        }
    }
}
```

# WCF LISTA PROVEEDORES

## IServicioListaProv.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;
using Dominio;

namespace WcfListaProv
{
    [ServiceContract]
    public interface IServicioListaProv
    {
        [OperationContract]
        IEnumerable<DtoProveedor> ObtenerProveedores();
    }

    [DataContract]
    public class DtoProveedor
    {
        [DataMember]
        public string RUT { get; set; }

        [DataMember]
        public string NombreFantasia { get; set; }
    }
}
```

## ServicioListaProv.svc

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;
using Dominio;

namespace WcfListaProv
{
    public class ServicioListaProv : IServicioListaProv
    {
        public IEnumerable<DtoProveedor> ObtenerProveedores()
        {
            List<Proveedor> listaCompleta = Proveedor.FindAll();
            if (listaCompleta == null) return null;
            List<DtoProveedor> proveedores = new List<DtoProveedor>();
            foreach (Proveedor p in listaCompleta)
            {
                proveedores.Add(
                    new DtoProveedor()
                    {
                        NombreFantasia = p.NombreFantasia,
                        RUT = p.RUT
                    }
                );
            }
            return proveedores;
        }
    }
}
```