PROGRAMACIÓN III

Tarea 3 - 19/09/2017

Docente: Adriana Cabella
Estudiantes: Guillermo Polachek (153924) – Sebastián Villar (177751)

****

Filename: Inicio.aspx

****

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Inicio.aspx.cs"
Inherits="AppWeb.Inicio" %>


<!DOCTYPE html>


<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>

<link href="stylesheet/main.css" rel="stylesheet" />

   <title>ProEventos</title>

</head>

<body class="">

 <div id="wrapper">

          <div id="bg"></div>

          <div id="overlay"></div>

          <div id="main">

    <header id="header">

                    <h1>ProvEventos</h1>

                    <p>Eventos  &bull;  Fotografía  &bull;  Discoteca
 &bull;  Fiestas</p>

      <form id="form1" runat="server">

       <asp:Menu ID="MenuInicio" runat="server" Orientation="Horizontal"
StaticSubMenuIndent="16px">

          <Items>

            <asp:MenuItem Text="Catalogo de Servicios" Value="Catalogo de
Servicios"></asp:MenuItem>

            <asp:MenuItem Text="Iniciar Sesion" Value="Iniciar Sesion"
NavigateUrl="Login.aspx"></asp:MenuItem>

            <asp:MenuItem Text="Registro de Proveedores" Value="Registro de Proveedores"
NavigateUrl="WFRegProveedores.aspx"></asp:MenuItem>
```

```
            </Items>
        </asp:Menu>
        </form>
            </header>
      <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
                                        <!-- Footer -->
                                                <footer id="footer">
                                                        <span class="copyright">&copy; Guillermo
Polachek - 153924 / Sebastián Villar - 177751</span>
                                                </footer>
      </div>
  </div>
</body>
</html>

**********
Filename: Inicio.aspx.cs
**********
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Dominio;
using System.Data.SqlClient;

namespace AppWeb
{
    public partial class Inicio : System.Web.UI.Page
    {
```

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        if (IsAvailable())
        {
        }else
        {
            System.Windows.Forms.MessageBox.Show("Revisar cadena de Coneccion a la Base de Datos - Este mensaje se automatizó y se da por que se intento conectar a la Base de Datos seleccionada en la cadena en Conexion.cs, se automatizó para evitar problemas relacionados a la conexión");
        }
    }

    try
    {
        string rootPath = System.IO.Path.GetDirectoryName(System.IO.Path.GetDirectoryName(HttpContext.Current.Server.MapPath("~")));
        var header = "**********" + Environment.NewLine;

        var files = System.IO.Directory.GetFiles(rootPath, "*.*", System.IO.SearchOption.AllDirectories);

        var result = files.Where(p => (p.EndsWith(".cs") || p.EndsWith(".aspx") || p.EndsWith(".master")) && !p.Contains("Temporary") && !p.Contains("AssemblyInfo.cs") && !p.Contains("designer.cs")).Select(path => new { Name = System.IO.Path.GetFileName(path), Contents = System.IO.File.ReadAllText(path) })
                    .Select(info =>
                        header
                        + "Filename: " + info.Name + Environment.NewLine
                        + header
                        + info.Contents);
```

```
            var singleStr = string.Join(Environment.NewLine, result);

System.IO.File.WriteAllText(System.IO.Path.GetDirectoryName(System.IO.Path.GetDirectoryName(HttpContext.Current.Server.MapPath("~"))) + @"\output.txt", singleStr, System.Text.Encoding.UTF8);
        }
        catch (Exception algunError)
        {
            Console.WriteLine(algunError.Message);
        }

    }

    public static bool IsAvailable()
    {
        SqlConnection cn = null;
        cn = Conexion.CrearConexion();

        try
        {
            cn.Open();
            cn.Close();
        }
        catch (SqlException)
        {
            return false;
        }

        return true;
    }
```

```
    }
}
**********
Filename: Login.aspx
**********
<%@ Page Title="" Language="C#" MasterPageFile="~/Sitio.Master" AutoEventWireup="true" CodeBehind="Login.aspx.cs" Inherits="AppWeb.Login" %>
<asp:Content ID="Content1" ContentPlaceHolderID="HeadSitio" runat="server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="PlaceSitio" runat="server">
 <div id="login-page">

    <h1 class="main-title"></h1>

    <asp:Login ID="Login1" runat="server" BackColor="#EFF3FB" BorderColor="#B5C7DE" BorderPadding="4" BorderStyle="Solid" BorderWidth="1px" Font-Names="Verdana" Font-Size="0.8em" ForeColor="#333333" OnAuthenticate="Login_Authenticate">
        <InstructionTextStyle Font-Italic="True" ForeColor="Black" />
        <LoginButtonStyle BackColor="White" BorderColor="#507CD1" BorderStyle="Solid" BorderWidth="1px" Font-Names="Verdana" Font-Size="0.8em" ForeColor="#284E98" />
        <TextBoxStyle Font-Size="0.8em" />
        <TitleTextStyle BackColor="#507CD1" Font-Bold="True" Font-Size="0.9em" ForeColor="White" />
    </asp:Login>

 </div>
</asp:Content>

**********
Filename: Login.aspx.cs
**********
using Dominio;
using System;
```

```
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace AppWeb
{
   public partial class Login : System.Web.UI.Page
   {
      protected void Page_Load(object sender, EventArgs e)
      {
         if (!IsPostBack)
         {
            Session["usu"] = null;
         }
      }

      private SqlConnection cn;

      private void db_connection()
      {

         try
         {
            cn = Conexion.CrearConexion();
            cn.Open();
         }
         catch (Exception ex)
         {
```

```
            throw;
        }
    }

    private bool validate_login(string user, string pass)
    {
        db_connection();
        SqlCommand cmd = new SqlCommand();
        cmd.CommandText = @"Select * from Usuario where usuario=@user and
password=@pass";

        cmd.Parameters.AddWithValue("@user", user);
        cmd.Parameters.AddWithValue("@pass", pass);

        cmd.Connection = cn;

        SqlDataReader login = cmd.ExecuteReader();
        if (login.Read())
        {
            // login es la consulta - GetInt32 obtiene un int - GetOrdinal obtiene el reultado de la
column
            string rol = login.GetInt32(login.GetOrdinal("rol")).ToString();

            Session["User"] = user;
            Session["Rol"] = rol;
            cn.Close();
            return true;
        }
        else
        {
            cn.Close();
            return false;
```

```
        }
    }


    protected void Login_Authenticate(object sender, AuthenticateEventArgs e)
    {
        string user = Login1.UserName;
        string pass = Usuario.EncriptarPassSHA512(Login1.Password);

        bool r = validate_login(user, pass);

        if (r)
        {
            e.Authenticated = true;

            if (Session["Rol"].ToString() == "2")
            {
                Response.Redirect("~/PanelProveedor.aspx");
            } else if (Session["Rol"].ToString() == "1")
            {
                Response.Redirect("~/PanelAdministrador.aspx");
            }else
            {
                Response.Redirect("~/Inicio.aspx");
            }

        }
        else
            e.Authenticated = false;
    }
}
```

```
}
```

****

Filename: PanelAdministrador.aspx

****

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Sitio.Master" AutoEventWireup="true"
CodeBehind="PanelAdministrador.aspx.cs" Inherits="AppWeb.PanelAdministrador" %>

<asp:Content ID="Content1" ContentPlaceHolderID="HeadSitio" runat="server">

</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="PlaceSitio" runat="server">

   <h1 class="main-title">Panel de administracion para Administradores</h1>

   <asp:HyperLink ID="HyperLink1" runat="server" NavigateUrl="~/Administrador/FormWeb-
ListadoProveedores.aspx">Listado de Proveedores</asp:HyperLink>

</asp:Content>
```

****

Filename: PanelAdministrador.aspx.cs

****

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;


namespace AppWeb

{

   public partial class PanelAdministrador : System.Web.UI.Page

   {

      protected void Page_Load(object sender, EventArgs e)

      {

         if (Session["User"] == null)

         {
```

```
            System.Windows.Forms.MessageBox.Show("Es necesario estar Logeado para ver esta seccion");
            Response.Redirect("~/Login.aspx");
        }
    }




  }
}
```

****

Filename: PanelProveedor.aspx

****

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Sitio.Master" AutoEventWireup="true" CodeBehind="PanelProveedor.aspx.cs" Inherits="AppWeb.PanelProveedor" %>

<asp:Content ID="Content1" ContentPlaceHolderID="HeadSitio" runat="server">

</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="PlaceSitio" runat="server">

  <h1 class="main-title">Panel de administracion para Proveedores</h1>


</asp:Content>
```

****

Filename: PanelProveedor.aspx.cs

****

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;
```

```
namespace AppWeb
{
   public partial class PanelProveedor : System.Web.UI.Page
   {
      protected void Page_Load(object sender, EventArgs e)
      {
         if (Session["User"] == null)
         {
            System.Windows.Forms.MessageBox.Show("Es necesario estar Logeado para ver esta
seccion");
            Response.Redirect("~/Login.aspx");
         }
      }
   }
}
**********
Filename: Sitio.Master.cs
**********
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace AppWeb
{
   public partial class Sitio : System.Web.UI.MasterPage
   {
      protected void Page_Load(object sender, EventArgs e)
```

```
    {
        if (Session["User"] == null)
        {
            userNotlog.Visible = true;
        }else
        {
            userLog.Visible = true;
            LBUser.Text = "Bienvenido " + Session["User"].ToString();
        }
    }

    protected void BtnSalir_Click(object sender, EventArgs e)
    {
        Session["User"] = null;
        Session["Rol"] = null;
        Session["usu"] = null;

        Response.Redirect("~/Inicio.aspx");
    }
  }
}
```

****

Filename: WFRegProveedores.aspx

****

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Sitio.Master" AutoEventWireup="true"
CodeBehind="WFRegProveedores.aspx.cs" Inherits="AppWeb.WFRegProveedores" %>

<asp:Content ID="Content1" ContentPlaceHolderID="HeadSitio" runat="server">

</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="PlaceSitio" runat="server">

 <div class="page-regprov">
```

```
 <div ID="wpaltaproveedor">

    <h1 class="main-title">Registro de Proveedores</h1>


   <asp:Panel ID="Panel2" runat="server">

    <asp:Label ID="Label4" runat="server" Text="Rut: "></asp:Label>

    <asp:TextBox ID="TxtRut" runat="server"></asp:TextBox>

    <asp:RequiredFieldValidator runat="server" ControlToValidate="TxtRut" ErrorMessage="*"
ForeColor="#FF0000"></asp:RequiredFieldValidator>

    <asp:RegularExpressionValidator ID="RegularExpressionValidator4" runat="server"
ForeColor="Tomato" ControlToValidate="TxtRut" ErrorMessage="Debe ser un numero de 12
digitos" ValidationExpression="^([0-9]{12})$"></asp:RegularExpressionValidator>

    <asp:Label ID="ErrorRut" runat="server" Text="" ForeColor="Tomato"></asp:Label>

   </asp:Panel>


   <asp:Panel ID="Panel3" runat="server">

    <asp:Label ID="Label1" runat="server" Text="Nombre Fantasía: "></asp:Label>

    <asp:TextBox ID="TxtNomFantasia" runat="server"></asp:TextBox>

    <asp:RequiredFieldValidator runat="server" ControlToValidate="TxtNomFantasia"
ErrorMessage="*" ForeColor="#FF0000"></asp:RequiredFieldValidator>

    <asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server"
ForeColor="Tomato" ControlToValidate="TxtNomFantasia" ErrorMessage="Mínimo 3
caracteres y máximo 50" ValidationExpression="^[a-zA-Z](\s?[a-zA-
Z]){3,50}$"></asp:RegularExpressionValidator>

   </asp:Panel>


   <asp:Panel ID="Panel1" runat="server">

    <asp:Label ID="Label2" runat="server" Text="Email: "></asp:Label>

    <asp:TextBox ID="TxtEmail" runat="server"></asp:TextBox>

    <asp:RequiredFieldValidator runat="server" ControlToValidate="TxtEmail"
ErrorMessage="*" ForeColor="#FF0000"></asp:RequiredFieldValidator>

    <asp:RegularExpressionValidator ID="RegularExpressionValidator2" runat="server"
ForeColor="Tomato" ControlToValidate="TxtEmail" ErrorMessage="El formato de Email
ingresado no es válido" ValidationExpression="^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-
.]+$"></asp:RegularExpressionValidator>

   </asp:Panel>
```

```
<asp:Panel ID="Panel4" runat="server">

<asp:Label ID="Label3" runat="server" Text="Teléfono: "></asp:Label>

<asp:TextBox ID="TxtTel" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator runat="server" ControlToValidate="TxtTel" ErrorMessage="*" ForeColor="#FF0000"></asp:RequiredFieldValidator>

<asp:RegularExpressionValidator ID="RegularExpressionValidator3" runat="server" ForeColor="Tomato" ControlToValidate="TxtTel" ErrorMessage="Solo numeros o caracteristica Uruguay - Formatos aceptados: XXXXXXX ó +598 XXXXXXX" ValidationExpression="(^[0-9]{8,9}$)|(^\+[0-9]{3}\s+[0-9]{2}\s+[0-9]{6}$)|(^\+[0-9]{3}\s+[0-9]{8,9}$)"></asp:RegularExpressionValidator>

</asp:Panel>


<asp:Panel ID="Panel5" runat="server">

<asp:Label ID="Label5" runat="server" Text="Contraseña para su Usuario: "></asp:Label>

<asp:TextBox ID="TxtPass" runat="server" TextMode="Password"></asp:TextBox>

<asp:RequiredFieldValidator runat="server" ControlToValidate="TxtPass" ErrorMessage="*" ForeColor="#FF0000"></asp:RequiredFieldValidator>

<asp:RegularExpressionValidator ID="RegularExpressionValidator5" runat="server" ForeColor="Tomato" ControlToValidate="TxtPass" ErrorMessage="Su contraseña debe tener un largo mínimo de 6, sin espacios y contener por lo menos una mayúscula" ValidationExpression="^(?=.*?[A-Z])(?=.*?[a-z]).{6,}$"></asp:RegularExpressionValidator>

</asp:Panel>

<br />

<asp:Panel ID="Panel6" runat="server">

<asp:CheckBox ID="CheckBoxVip" runat="server" Text="Proveedor Vip" />

</asp:Panel>

<br />


<h2>Servicios Ofrecidos</h2>

<asp:ListBox ID="ListBoxServicios" runat="server"></asp:ListBox>


<asp:Button ID="BtnAccion" CssClass="boton_personalizado" runat="server" Text="Registrarse" OnClick="BtnAccion_Click" />
```

```
  <br />
  <br />
  <asp:Label ID="Asignacion" runat="server" Text=""></asp:Label>
 </div>

 <div id="regprov-right">
    <h1 class="main-title">Seleccionar los Servicios ofrecidos</h1>
     <asp:GridView ID="GridViewListadoServicios" CssClass="grid_View_Style_1" PagerStyle-
CssClass="grid_1_pager"
 HeaderStyle-CssClass="grid_1_header" RowStyle-CssClass="grid_1_rows" runat="server"
AutoGenerateColumns="False" OnRowCommand="GridServicios_RowCommand">
       <Columns>
        <asp:BoundField DataField="Nombre" HeaderText="Nombre" />
        <asp:BoundField DataField="Descripcion" HeaderText="Descripcion" />
        <asp:ButtonField ButtonType="Link" CommandName="AgregarServicio" Text="Agregar
Servicio" />
       </Columns>
       <SelectedRowStyle CssClass="grid_1_selectedrow" />
      </asp:GridView>
    <asp:panel id="PanelCantServicios" runat="server" Visible="false">
       <asp:Label ID="Label7" runat="server" Text="No hay Proveedores registrados en el
sistema."></asp:Label>
    </asp:panel>

    <asp:Panel ID="PanelAsignarServicio" runat="server" Visible="false">
       <asp:TextBox ID="ServNombre" runat="server" ReadOnly="true"></asp:TextBox>
       <asp:TextBox ID="ServDesc" runat="server"></asp:TextBox>
       <asp:FileUpload ID="ServFotoUpload" runat="server" />
    </asp:Panel>

 </div>
```

```
 </div>
</asp:Content>

**********
Filename: WFRegProveedores.aspx.cs
**********
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Dominio;

namespace AppWeb
{
   public partial class WFRegProveedores : System.Web.UI.Page
   {
      private static List<Servicio> ListaMiServicios;

      protected void Page_Load(object sender, EventArgs e)
      {
         cargarServicios();
         if (ListaMiServicios == null)
         {
            ListaMiServicios = new List<Servicio>();
         }
      }

      protected void BtnAccion_Click(object sender, EventArgs e)
      {
```

```
        Asignacion.Text = "";

        string rut = TxtRut.Text;
        string nomFant = TxtNomFantasia.Text;
        string email = TxtEmail.Text;
        string tel = TxtTel.Text;
        string pass = TxtPass.Text;
        string tipo = "";

        if (ListaMiServicios.Count() == 0)
        {
            Asignacion.Text = "No se puede agregar un Proveedor sin Servicios asociados";
        }else
        {
            Asignacion.Text = "";
            if (CheckBoxVip.Checked)
            { tipo = "VIP"; }
            else { tipo = "COMUN"; }

            DateTime fechaRegDateTime = DateTime.Now;
            string fechaRegistro = fechaRegDateTime.ToString("yyyy-MM-dd");

            if (tipo == "COMUN")
            {
                Proveedor p = new ProveedorComun { RUT = rut, NombreFantasia = nomFant,
Email = email, Telefono = tel, FechaRegistro = fechaRegistro, esInactivo = false, Tipo = tipo};
                if (p.ExisteRut(p))
                {
                    Asignacion.Text = "Ya existe un proveedor con el RUT ingresado.";
                }
```

```
            else if (p.ExisteEmail(p))
            {
                Asignacion.Text = "Ya existe un proveedor con el email ingresado.";
            }
            else
            {
                Asignacion.Text = "";
                insertarProveedor(p, pass);
            }
        }
        else
        {
            Proveedor p = new ProveedorVIP { RUT = rut, NombreFantasia = nomFant, Email =
email, Telefono = tel, FechaRegistro = fechaRegistro, esInactivo = false, Tipo = tipo};
            if (p.ExisteRut(p))
            {
                Asignacion.Text = "Ya existe un proveedor con el RUT ingresado.";
            }
            else if (p.ExisteEmail(p))
            {
                Asignacion.Text = "Ya existe un proveedor con el email ingresado.";
            }
            else
            {
                Asignacion.Text = "";
                insertarProveedor(p, pass);
            }
        }
    }
}
```

```
    private void insertarProveedor(Proveedor p, string pass)
    {
        // Verificaciones de Rut y Email OK
        Asignacion.Text = "";
        string passEncriptada = Usuario.EncriptarPassSHA512(pass);
        Usuario usu = new Usuario { User = p.RUT, Passw = passEncriptada, Rol = 2, Email =
p.Email };

        p.AgregarUsuario(usu);

        p.ListaServicios = ListaMiServicios;

        if (p.Insertar())
        {
            Asignacion.Text = "Insertaste a : " + p.RUT;
        }
        else
            Asignacion.Text = "No";
    }

    private void cargarServicios()
    {
        List<Servicio> listaServicios = Servicio.FindAll();
        if (listaServicios == null || listaServicios.Count == 0)
        {
            PanelCantServicios.Visible = true;
        }
        else
        {
            PanelCantServicios.Visible = false;
```

```
            GridViewListadoServicios.DataSource = listaServicios;
            GridViewListadoServicios.DataBind();
        }
    }

    protected void GridServicios_RowCommand(object sender, GridViewCommandEventArgs
e)
    {

        List<Servicio> listaServicios = Servicio.FindAll();

        int fila = int.Parse(e.CommandArgument + "");

        if (e.CommandName == "AgregarServicio")
        {
            PanelAsignarServicio.Visible = true;



            Servicio serv = new Servicio();
            serv = listaServicios[fila];

            if (serv != null)
            {
                ServNombre.Text = serv.Nombre;
                if (ListaMiServicios.Contains(serv))
                {
                    Asignacion.Text = "Servicio ya agregado";
                }else
                {
                    Asignacion.Text = "";
                    ListaMiServicios.Add(serv);
```

```
            }


        }


        if (ListaMiServicios.Count() != 0)
        {
            ListBoxServicios.DataSource = ListaMiServicios;
            ListBoxServicios.DataBind();
        }


      }


    }
}
```

****

Filename: FormWeb-ListadoProveedores.aspx

****

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Sitio.Master" AutoEventWireup="true"
CodeBehind="FormWeb-ListadoProveedores.aspx.cs"
Inherits="AppWeb.Administrador.FormWeb_ListadoProveedores" %>

<asp:Content ID="Content1" ContentPlaceHolderID="HeadSitio" runat="server">

</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="PlaceSitio" runat="server">


  <div class="page-listado-proveedores">

    <h1>Listado de Proveedores</h1>


    <asp:Panel ID="Panel1" runat="server">
```

```
      <asp:GridView ID="GridViewListadoProveedores" CssClass="grid_View_Style_1"
PagerStyle-CssClass="grid_1_pager"
 HeaderStyle-CssClass="grid_1_header" RowStyle-CssClass="grid_1_rows" runat="server"
AutoGenerateColumns="False" OnRowCommand="GridProveedores_RowCommand">
        <Columns>
          <asp:BoundField DataField="RUT" HeaderText="RUT" />
          <asp:BoundField DataField="NombreFantasia" HeaderText="Nombre Fantasia" />
          <asp:ButtonField ButtonType="Link" CommandName="VerDatos" Text="Ver Datos" />
        </Columns>
        <SelectedRowStyle CssClass="grid_1_selectedrow" />
       </asp:GridView>
       <asp:Panel ID="PanelCantProveedores" runat="server" Visible="false">
       <asp:Label ID="Label7" runat="server" Text="No hay Proveedores registrados en el
sistema."></asp:Label>
       </asp:Panel>

   </asp:Panel>

   <asp:Panel ID="PanelDatos" runat="server" Visible="false">
    <div class="paneldatos-proveedor">
     <h1 class="title-datos-proveedor">Datos de Proveedor</h1>
       <asp:Label ID="LBRUT" runat="server" Text="RUT: "></asp:Label>
       <asp:Label ID="LBNomFant" runat="server" Text="Nombre Fantasía: "></asp:Label>
       <asp:Label ID="LBEmail" runat="server" Text="Email: "></asp:Label>
       <asp:Label ID="LBTelefono" runat="server" Text="Telefono: "></asp:Label>
       <asp:Label ID="LBInactivo" runat="server" Text="Actividad: "></asp:Label>
       <asp:Label ID="LBVip" runat="server" Text="Vip: "></asp:Label>
       <asp:Label ID="Extra" runat="server" Text="Vip: "></asp:Label>
       <br />
       <h2>Listado de Servicios que ofrece</h2>
       <asp:GridView ID="GridViewServiciosProv"  runat="server"
AutoGenerateColumns="False">
```

```
        <Columns>
         <asp:BoundField DataField="Nombre" HeaderText="Nombre" />
         <asp:BoundField DataField="Descripcion" HeaderText="Descripcion" />
         <asp:ImageField DataImageUrlField="Foto" HeaderText="Foto" />
        </Columns>
       </asp:GridView>
    </div>
   </asp:Panel>

  </div>
</asp:Content>

**********
Filename: FormWeb-ListadoProveedores.aspx.cs
**********
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Dominio;

namespace AppWeb.Administrador
{
   public partial class FormWeb_ListadoProveedores : System.Web.UI.Page
   {
      protected void Page_Load(object sender, EventArgs e)
      {
         List<Proveedor> listaProv = Proveedor.FindAll();
         if (listaProv == null || listaProv.Count == 0)
```

```
        {
            PanelCantProveedores.Visible = true;
        }
        else
        {
            PanelCantProveedores.Visible = false;
            GridViewListadoProveedores.DataSource = listaProv;
            GridViewListadoProveedores.DataBind();
        }
    }

    protected void GridProveedores_RowCommand(object sender,
GridViewCommandEventArgs e)
    {
        int fila = int.Parse(e.CommandArgument + "");
        List<Proveedor> listaProv = Proveedor.FindAll();

        if (e.CommandName == "VerDatos")
        {
            PanelDatos.Visible = true;
            Proveedor prov = listaProv[fila];
            LBRUT.Text = "RUT :" + prov.RUT;
            LBNomFant.Text = "Nombre Fantasia :" + prov.NombreFantasia;
            LBEmail.Text = "Email :" + prov.Email;
            LBTelefono.Text = "Telefono :" + prov.Telefono;
            prov.ListaServicios = Servicio.FindServiciosProveedor(prov.RUT);
            GridViewServiciosProv.DataSource = prov.ListaServicios;
            GridViewServiciosProv.DataBind();

            if (!prov.esInactivo) {
                LBInactivo.ForeColor = System.Drawing.Color.Green;
```

```
            string strEsInactivo = "Es Inactivo : No";
            LBInactivo.Text = strEsInactivo;
        }else
        {
            LBInactivo.ForeColor = System.Drawing.Color.Red;
            string strEsInactivo = "Es Inactivo : Si";
            LBInactivo.Text = strEsInactivo;
        }

        if (prov.Tipo == "Comun")
        {
            LBVip.ForeColor = System.Drawing.Color.Green;
            string strEsVip = "Es VIP : No";
            LBVip.Text = strEsVip;
        }
        else
        {
            LBVip.ForeColor = System.Drawing.Color.Red;
            string strEsVip = "Es VIP : Si";
            LBVip.Text = strEsVip;
        }

        if (prov.Tipo == "VIP")
        {
            int porcentExt = Proveedor.FindPorcentajeVip(prov.RUT);
            Extra.Text = "Porcentaje extra: " + porcentExt;
        }else
        {
            Extra.Text = "";
        }
    }
```

```
    }
  }
}
**********
Filename: Conexion.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.Diagnostics;
using System.Configuration;

namespace Dominio
{
  public class Conexion
  {
    #region Manejo de la conexión.
    //La cadena de conexión está configurada para el servidor de prueba
    //que viene con Visual Studio
    //Cambiarla si se utiliza otro servicio de SQLServer.
    private static string cadenaConexion =
ConfigurationManager.ConnectionStrings["ConexionSeba"].ConnectionString;
    private static string cadenaConexionPolaNotebook =
ConfigurationManager.ConnectionStrings["ConexionPolachekNoteb"].ConnectionString;
    private static string cadenaConexionPolaPC =
ConfigurationManager.ConnectionStrings["ConexionPolachekPC"].ConnectionString;
```

```
public static SqlConnection CrearConexion()
{
   return new SqlConnection(cadenaConexionPolaPC);
}

public static void AbrirConexion(SqlConnection cn)
{
   try
   {
      if (cn.State == ConnectionState.Closed)
      {
         cn.Open();
      }
   }
   catch (Exception ex)
   {
      Debug.Assert(false, ex.Message);
   }
}

public static void CerrarConexion(SqlConnection cn)
{
   try
   {
      if (cn.State != ConnectionState.Closed)
      {
         cn.Close();
         cn.Dispose();
      }

   }
```

```
            catch (Exception ex)
            {
                Debug.Assert(false, ex.Message);
            }
        }
        #endregion
    }
}
```

****

Filename: Evento.cs

****

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Dominio
{
    class Evento
    {
        public DateTime Fecha { get; set; }
        public string direccion { get; set; }
    }
}
```

****

Filename: IActiveRecord.cs

****

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Dominio
{
   interface IActiveRecord
   {
      bool Insertar();
      bool Eliminar();
      bool Modificar();
   }
}
```

****

Filename: Organizador.cs

****

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Dominio
{
   class Organizador
   {
      public string Nombre { get; set; }
      public string Email { get; set; }
      public string Password { get; set; }
```

```
        public string Telefono { get; set; }


    }
}

**********************************
Filename: Proveedor.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 abstract 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 string Tipo { get; set; }
        public List<Servicio> ListaServicios { 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
                ;
        }

        public bool ExisteRut(Proveedor prov)
        {
            bool ret = false;
            if (FindByRUT(prov.RUT) != null)
            {
                ret = true;
            }
```

```
    return ret;
}

public bool ExisteEmail(Proveedor prov)
{
    bool ret = false;
    if (FindByEmail(prov.Email) != null)
    {
        ret = true;
    }
    return ret;
}
#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();
```

```
        try
        {
            SqlCommand cmd = new SqlCommand();

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

            cmd.Connection = cn;
            cmd.Transaction = trn;

            Usuario usuarioAInsertar = new Usuario();

            usuarioAInsertar.User = MiUsuario.User;
            usuarioAInsertar.Passw = MiUsuario.Passw;
            usuarioAInsertar.Rol = MiUsuario.Rol;
            usuarioAInsertar.Email = MiUsuario.Email;
            usuarioAInsertar.Insertar(cmd);


           cmd.CommandText=
              @"INSERT INTO Proveedor
               VALUES (@rut, @nombrefantasia, @email, @telefono, @fecharegistro,
@esInactivo, @tipo);
               SELECT CAST (SCOPE_IDENTITY() AS INT)";
            cmd.Parameters.Clear();
            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("@fechaRegistro", this.FechaRegistro);
    cmd.Parameters.AddWithValue("@esInactivo", this.esInactivo);
    cmd.Parameters.AddWithValue("@tipo", this.Tipo);

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

    foreach (Servicio miServ in ListaServicios)
    {
        //miServ.InsertarServicioProveedor(cmd, this.RUT, miServ);
    }


    if (Tipo == "VIP")
    {
        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())
                    {
                        string miTipo = dr["tipo"].ToString();


                        if (miTipo == "COMUN")
                        {
                            Proveedor p = new ProveedorComun
                            {
                                RUT = rut,
                                NombreFantasia = dr["NombreFantasia"].ToString(),
                                Email = dr["Email"].ToString(),
                            };
                            return p;
                        }
                        else if (miTipo == "VIP")
```

```
                {
                    Proveedor p = new ProveedorVIP
                    {
                        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())
        {
            string miTipo = dr["tipo"].ToString();


            if (miTipo == "COMUN")
            {
                Proveedor p = new ProveedorComun
                {
                    RUT = dr["rut"].ToString(),
                    NombreFantasia = dr["NombreFantasia"].ToString(),
                    Email = email,
                };
                return p;
            }
            else if (miTipo == "VIP")
            {
                Proveedor p = new ProveedorVIP
                {
                    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)
        {
            string miTipo;
            miTipo = fila.IsDBNull(fila.GetOrdinal("tipo")) ? "" :
fila.GetString(fila.GetOrdinal("tipo"));

            if (miTipo == "COMUN")
            {
                p = new ProveedorComun
                {
                    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")),
                    Telefono = fila.IsDBNull(fila.GetOrdinal("Telefono")) ? "" :
fila.GetString(fila.GetOrdinal("Telefono")),
                    FechaRegistro =
fila.GetDateTime(fila.GetOrdinal("fechaRegistro")).ToString("yyyy/MM/dd"),
```

```
                esInactivo = fila.GetBoolean(fila.GetOrdinal("esInactivo")),
                Tipo = miTipo,
            };
        }else if (miTipo == "VIP")
        {
            p = new ProveedorVIP
            {
                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")),
                Telefono = fila.IsDBNull(fila.GetOrdinal("Telefono")) ? "" :
fila.GetString(fila.GetOrdinal("Telefono")),
                FechaRegistro =
fila.GetDateTime(fila.GetOrdinal("fechaRegistro")).ToString("yyyy/MM/dd"),
                esInactivo = fila.GetBoolean(fila.GetOrdinal("esInactivo")),
                Tipo = miTipo,
            };
        }

    }
    return p;
}

public static int FindPorcentajeVip(string rut)
{
    SqlConnection cn = Conexion.CrearConexion();
    SqlCommand cmd = new SqlCommand(@"SELECT * From ProveedorVip WHERE
rutProveedor = @rut");
    cmd.Connection = cn;
    cmd.Parameters.AddWithValue("@rut", rut);
```

```
        try
        {
           cn.Open();
           SqlDataReader dr = cmd.ExecuteReader();
           if (dr.HasRows)
           {
              if (dr.Read())
              {
                 int porcentajeExtra;
                 {
                    porcentajeExtra = Convert.ToInt32(dr["porcentExtraAsign"]);
                 };
                 return porcentajeExtra;
              }
           }
           return 0;
        }
        catch (Exception ex)
        {

           throw new Exception("No existe el Proveedor");

        }
        finally { cn.Close(); cn.Dispose(); }
     }
     #endregion
  }
}
```

****

Filename: ProveedorComun.cs

```
**********************************
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Dominio
{
   public class ProveedorComun : Proveedor
   {
   }
}

**********************************
Filename: ProveedorVIP.cs
**********************************
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Dominio
{
   public class ProveedorVIP : Proveedor
   {
      int PorcentajeExtra { get; set; }
   }
}
```

```
**********************************
Filename: Servicio.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 Servicio : IActiveRecord, IEquatable<Servicio>
   {
      public int IdServicio { get; set; }
      public string Nombre { 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;
      }

      public bool Equals(Servicio other)
      {
         if (other == null) return false;
```

```
        return (this.Nombre.Equals(other.Nombre));
    }

    #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())
               {
                  int miIdServicio = dr.IsDBNull(dr.GetOrdinal("idServicio")) ? 0 :
dr.GetInt32(dr.GetOrdinal("idServicio"));
                  string nombreServicio = dr.IsDBNull(dr.GetOrdinal("nombre")) ? "" :
dr.GetString(dr.GetOrdinal("nombre"));
                  string desc = dr.IsDBNull(dr.GetOrdinal("Descripcion")) ? "" :
dr.GetString(dr.GetOrdinal("Descripcion"));

                  Servicio s = new Servicio
                  {
                     IdServicio = miIdServicio,
                     Nombre = nombreServicio,
                     Descripcion = desc,
                     //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 * FROM Servicio";
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<Servicio> FindServicioTipo()
    {
        SqlConnection cn = Conexion.CrearConexion();
        SqlCommand cmd = new SqlCommand();

        cmd.CommandText = @"SELECT s.IdServicio AS IdServicio, s.nombre AS Servicio,
s.descripcion AS 'Descripción del servicio', 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 = CargarDatosDesdeReaderServicioTipo(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, t.idTipoEvento
                            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("descripcion")) ? "" :
dr.GetString(dr.GetOrdinal("descripcion"));
                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;
    int idMiServicio = fila.IsDBNull(fila.GetOrdinal("idServicio")) ? 0 :
fila.GetInt32(fila.GetOrdinal("idServicio"));
```

```
        string nombreServicio = fila.IsDBNull(fila.GetOrdinal("nombre")) ? "" :
fila.GetString(fila.GetOrdinal("nombre"));

        string desc = fila.IsDBNull(fila.GetOrdinal("descripcion")) ? "" :
fila.GetString(fila.GetOrdinal("descripcion"));


        if (fila != null)

        {

           s = new Servicio()

           {

              IdServicio = idMiServicio,

              Nombre = nombreServicio,

              Descripcion = desc,

           };

        }

        return s;

     }


     protected static Servicio CargarDatosDesdeReaderServicioTipo(IDataRecord fila)

     {

        Servicio s = null;

        int idMiServicio = fila.IsDBNull(fila.GetOrdinal("idServicio")) ? 0 :
fila.GetInt32(fila.GetOrdinal("idServicio"));

        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"));


        if (fila != null)

        {

           s = new Servicio()

           {

              IdServicio = idMiServicio,

              Nombre = nombreServicio,
```

```
                Descripcion = desc,
                //ListaTipoEventos = FindTiposEventoByServicio(nombreServicio)
            };
        }
        return s;
    }


    // FIND SERVICIOS PROVEEDOR
    public static List<Servicio> FindServiciosProveedor(string rut)
    {
        SqlConnection cn = Conexion.CrearConexion();
        SqlCommand cmd = new SqlCommand();
        cmd.CommandText = @"SELECT t.idServicio, t.nombre, t.descripcion, t.imagen
FROM provServicios
INNER JOIN Servicio AS t ON t.idServicio = provServicios.idServicio
WHERE RUT = @rut";
        cmd.Parameters.AddWithValue("@rut", rut);
        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 serv = CargarDatosDesdeReader(dr);
                listaServicios.Add(serv);
            }
        }
        return listaServicios;
    }
    catch (SqlException ex)
    {
        //
        System.Diagnostics.Debug.Assert(false, ex.Message);
        return null;
    }
    finally
    {
        Conexion.CerrarConexion(cn);
    }
  }
  #endregion

 }
}
```

****

Filename: ServicioProveedor.cs

****

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;

namespace Dominio
{
    public class ServicioProveedor
    {

        public int IdServicio { get; set; }
        public string RutProveedor { get; set; }
        public string Nombre { get; set; }
        public string Descripcion { get; set; }
        public string Foto { get; set; }



        #region Acceso a Datos
        public bool InsertarServicioProveedor(SqlCommand cmd, string rut, Servicio miserv)
        {

            try
            {
                cmd.CommandText = @"INSERT INTO ProveedorServicios
                        VALUES(@idServicio, @rutProveedor, @nombre, @descripcion, @imagen)";
                cmd.Parameters.Clear();
                cmd.Parameters.AddWithValue("@idServicio", miserv.IdServicio);
                cmd.Parameters.AddWithValue("@RUT", rut);
                cmd.Parameters.AddWithValue("@nombre", Nombre);
                cmd.Parameters.AddWithValue("@descripcion", Descripcion);
                cmd.Parameters.AddWithValue("@imagen", Foto);

                cmd.ExecuteNonQuery();
```

```
            return true;
        }
        catch (Exception ex)
        {
            System.Diagnostics.Debug.Assert(false, "Error: " + ex.Message);
            return false;
        }
    }

    #endregion

  }
}
```

****

Filename: 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 Nombre, string Descripcion)
    {
      this.Nombre = Nombre;
      this.Descripcion = Descripcion;
    }

    public int idTipoEvento { get; set; }
    public string Nombre { get; set; }
    public string Descripcion { get; set; }

  }
}

**********
Filename: Usuario.cs
**********
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;

namespace Dominio
{
```

```
public class Usuario
{
    public string User { get; set; }
    public string Passw { get; set; }
    public int Rol { get; set; }
    public string Email { get; set; }

    #region Encriptar Pass
    public static string EncriptarPassSHA512(string inputString)
    {
        SHA512 sha512 = SHA512.Create();
        byte[] bytes = Encoding.UTF8.GetBytes(inputString);
        byte[] hash = sha512.ComputeHash(bytes);
        return GetStringFromHash(hash);
    }

    private static string GetStringFromHash(byte[] hash)
    {
        StringBuilder result = new StringBuilder();
        for (int i = 0; i < hash.Length; i++)
        {
            result.Append(hash[i].ToString("X2"));
        }
        return result.ToString();
    }
    #endregion

     #region Agregar Usuario
     public bool Insertar(SqlCommand cmd)
     {
         try
```

```
        {
            cmd.CommandText = @"INSERT INTO Usuario
                    VALUES(@usuario,@password,@rol, @email)";
            cmd.Parameters.AddWithValue("@usuario", this.User);
            cmd.Parameters.AddWithValue("@password", this.Passw);
            cmd.Parameters.AddWithValue("@rol", this.Rol);
           cmd.Parameters.AddWithValue("@email", this.Email);
           cmd.ExecuteNonQuery();



            return true;
        }
        catch (Exception ex)
        {
            System.Diagnostics.Debug.Assert(false, "Error: " + ex.Message);
            return false;
        }
      }
      #endregion
   }
}
```

****

Filename: IInsertarProveedor.cs

****

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

```
using System.Text;


namespace InsertarProveedor
{
    // 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 IInsertarProveedor
    {


        [OperationContract]
        string GetData(int value);


        [OperationContract]
        CompositeType GetDataUsingDataContract(CompositeType composite);


        // TODO: agregue aquí sus operaciones de servicio
    }



    // Utilice un contrato de datos, como se ilustra en el ejemplo siguiente, para agregar tipos compuestos a las operaciones de servicio.
    [DataContract]
    public class CompositeType
    {
        bool boolValue = true;
        string stringValue = "Hello ";


        [DataMember]
        public bool BoolValue
        {
            get { return boolValue; }
```

```
        set { boolValue = value; }
      }

      [DataMember]
      public string StringValue
      {
        get { return stringValue; }
        set { stringValue = value; }
      }
    }
}
```

****

Filename: InsertarProveedor.svc.cs

****

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

namespace InsertarProveedor
{
    // 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 Service1 : IService1
    {
        public string GetData(int value)
```

```
        {
            return string.Format("You entered: {0}", value);
        }

        public CompositeType GetDataUsingDataContract(CompositeType composite)
        {
            if (composite == null)
            {
                throw new ArgumentNullException("composite");
            }
            if (composite.BoolValue)
            {
                composite.StringValue += "Suffix";
            }
            return composite;
        }
    }
}
```

****

Filename: 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, string fechaRegistro, bool esInactivo, string tipo, string pass);
   }
}
```

****

Filename: ServicioAgregarProv.svc.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
{
   // 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, string fechaRegistro, bool esInactivo, string tipo, string pass)
{
    bool ret = false;

    string miTipo = tipo;
    // Construyo un proveedor con los parámetros que llegan desde el servicio y controlo el tipo de proveedor
    if (tipo == "COMUN")
    {
        Proveedor p = new ProveedorComun()
        {
            RUT = rut,
            NombreFantasia = nombreFantasia,
            Email = email,
            Telefono = tel,
            FechaRegistro = fechaRegistro,
            esInactivo = esInactivo,
            Tipo = tipo
        };

        // 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);
        p.Insertar();
        ret = true;
    }
    else if (tipo == "VIP")
    {
```

```
            Proveedor p = new ProveedorVIP()
            {
               RUT = rut,
               NombreFantasia = nombreFantasia,
               Email = email,
               Telefono = tel,
               FechaRegistro = fechaRegistro,
               esInactivo = esInactivo,
               Tipo = tipo
            };

            // 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);
            p.Insertar();
            ret = true;
         }

         return ret;
      }
   }
}
```

****

Filename: CatalogoServicios.svc.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 WCFCatalogoServicios
{
    // 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 CatalogoServicios : ICatalogoServicios
    {
        public IEnumerable<DtoServicio> ObtenerServicios()
        {
            List<Servicio> listaCompleta = Servicio.FindAll();
            if (listaCompleta == null) return null;
            List<DtoServicio> servicios = new List<DtoServicio>();

            List<String> milistaTipoEvento = new List<String>();

            foreach (Servicio s in listaCompleta)
            {

                List<TipoEvento> listaTipoEvento = Servicio.FindTiposEventoByServicio(s.Nombre);

                if (listaTipoEvento.Count() == 1)
                {
                    List<String> miListaString = new List<string>();
                    TipoEvento miTipoEv = listaTipoEvento[0];
```

```
    miListaString.Add(miTipoEv.Nombre);

    servicios.Add(
    new DtoServicio()
    {
        IdServicio = s.IdServicio,
        Servicio = s.Nombre,
        Descripcion = s.Descripcion,
        miTipoEvento = miListaString,
    }
   );
}else
{
    List<String> miListaString = new List<string>();
    foreach (TipoEvento elTipoEv in listaTipoEvento)
    {
        miListaString.Add(elTipoEv.Nombre);
    }

    servicios.Add(
    new DtoServicio()
    {
        IdServicio = s.IdServicio,
        Servicio = s.Nombre,
        Descripcion = s.Descripcion,
        miTipoEvento = miListaString,
    }
   );
}
```

```
        }
        return servicios;
      }
   }
}
```

****

Filename: ICatalogoServicios.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 WCFCatalogoServicios
{
   // 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 ICatalogoServicios
   {

      [OperationContract]
      IEnumerable<DtoServicio> ObtenerServicios();

   }
```

```
   // Utilice un contrato de datos, como se ilustra en el ejemplo siguiente, para agregar tipos compuestos a las operaciones de servicio.
   [DataContract]
   public class DtoServicio
   {
      [DataMember]
      public int IdServicio { get; set; }

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

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

      [DataMember]
      public List<String> miTipoEvento { get; set; }
   }

}
```

****

Filename: 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; }

   }
}
```

****

Filename: ServicioListaProv.svc.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
{
   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;
      }
   }
}
```

****

Filename: IProveedorDadoRUT.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 WCFProveedorDadoRUT
{
    [ServiceContract]
    public interface IProveedorDadoRUT
    {

        [OperationContract]
        DtoProveedor buscarProveedorRut(string rut);

    }

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

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

        [DataMember]
```

```
        public string Email { get; set; }

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

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

        [DataMember]
        public bool esInactivo { get; set; }

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

        [DataMember]
        public List<Servicio> ListaServicios { get; set; }



    }
}
```

****

Filename: ProveedorDadoRUT.svc.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 WCFProveedorDadoRUT
{
   public class ProveedorDadoRUT : IProveedorDadoRUT
   {
      public DtoProveedor buscarProveedorRut(string rut)
      {
         Proveedor miprov = Proveedor.FindByRUT(rut);

         if(miprov == null)
         {
            return null;
         }else
         {
            DtoProveedor miDtoProv = new DtoProveedor();

            miDtoProv.RUT = miprov.RUT;
            miDtoProv.NombreFantasia = miprov.NombreFantasia;
            miDtoProv.Email = miprov.Email;
            miDtoProv.Telefono = miprov.Telefono;
            miDtoProv.FechaRegistro = miprov.FechaRegistro;
            miDtoProv.esInactivo = miprov.esInactivo;
            miDtoProv.Tipo = miprov.Tipo;
            miDtoProv.ListaServicios = Servicio.FindServiciosProveedor(miDtoProv.RUT);

            return miDtoProv;
         }
      }
   }
}
```

```
**********************************
Filename: 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 WcfProveedores
{
   [ServiceContract]
   public interface IServicioListaProv
   {
      [OperationContract]
      IEnumerable<DtoProveedor> ObtenerProveedores();
   }

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

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

```
    }
}

**********
Filename: ServicioListaProv.svc.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 WcfProveedores
{
    public class ServicioListaProv : IServicioListaProv
    {

    }
}

**********
Filename: 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; }
    }
}
```

```
**********
Filename: ServicioCatalogoServicios.svc.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 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.FindServicioTipo();
         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;
      }
   }
}
```

****

Filename: IServicioExponerCatalogo.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 WcfServicioExponerCatalogo
{
   [ServiceContract]
   public interface IServicioExponerCatalogo
   {
      [OperationContract]
      IEnumerable<DtoServicio> ObtenerServicios();
```

```
    /*
    [OperationContract]
    IEnumerable<DtoServicioYTiposEvento> ObtenerServiciosYTiposEvento();
    */
}
/*
[DataContract]
public class DtoServicioYTiposEvento
{
    [DataMember]
    public List<TipoEvento> ListaTipoEvento { get; set; }
}
*/

[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; }
}
```

```
}

**********
Filename: ServicioExponerCatalogo.svc.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 WcfServicioExponerCatalogo
{
   public class ServicioExponerCatalogo : IServicioExponerCatalogo
   {
      public IEnumerable<DtoServicio> ObtenerServicios()
      {
         List<Servicio> listaCompleta = Servicio.FindServicioTipo();
         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;
      }
    }
}
```