

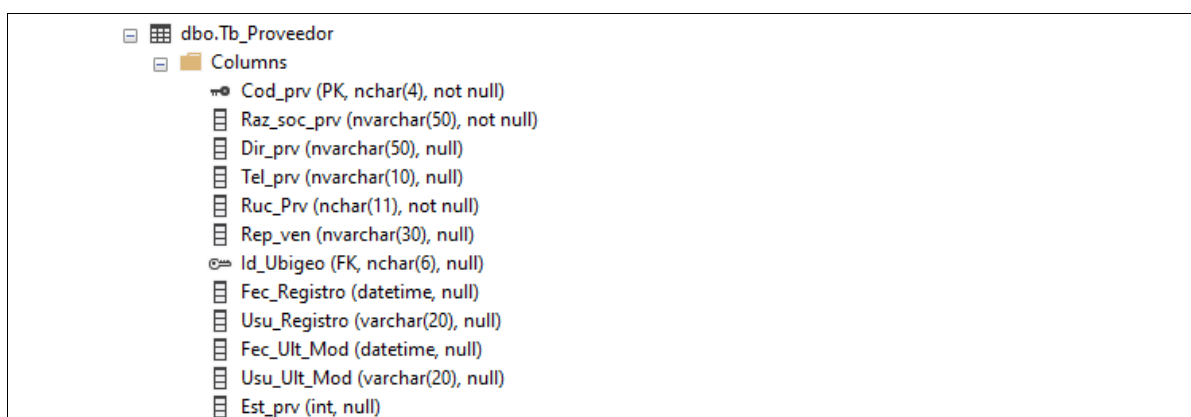
GUIA PRÁCTICA SEMANA 06

Competencia:

El desarrollo de esta guía práctica te permitirá conocer iniciar el mantenimiento de la tabla Tb_Proveedor, como parte del tu proceso de aprendizaje

Especificación Técnica:

La guía se basa en la solución **DemoVentas_3Capas** desarrollada desde la sesión 5 y la base de datos **VentasLeon**. La tabla **Tb_Proveedor** tiene la siguiente estructura:



The screenshot shows the 'Columns' list for the 'dbo.Tb_Proveedor' table. The columns and their data types are as follows:

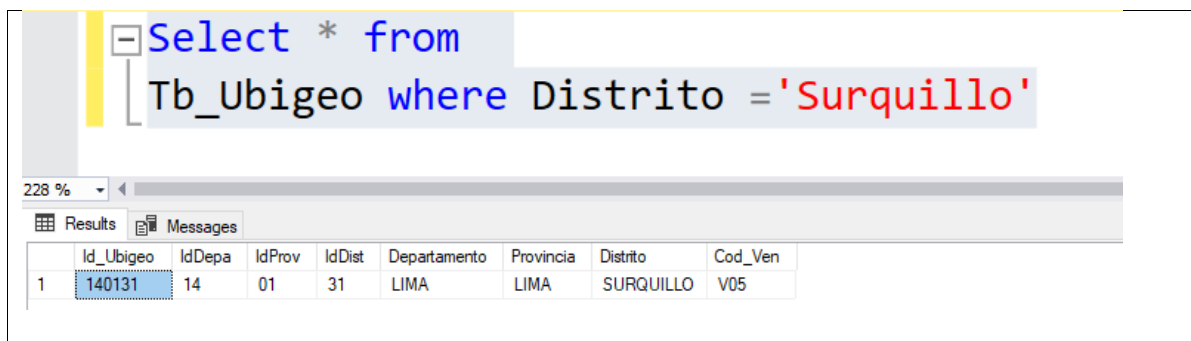
Column Name	Data Type	Constraints
Cod_prv	nchar(4)	PK, not null
Raz_soc_prv	nvarchar(50)	not null
Dir_prv	nvarchar(50)	null
Tel_prv	nvarchar(10)	null
Ruc_Priv	nchar(11)	not null
Rep_ven	nvarchar(30)	null
Id_Ubigeo	nchar(6)	FK, null
Fec_Registro	datetime	null
Usu_Registro	varchar(20)	null
Fec_Ult_Mod	datetime	null
Usu_Ult_Mod	varchar(20)	null
Est_prv	int	null

Para el mantenimiento de esta tabla, emplearemos los siguientes procedimientos almacenados:

- **usp_InsertarProveedor:** Que permitirá insertar un nuevo proveedor. Este procedimiento almacenado genera el código del siguiente proveedor, por lo que no es necesario ingresarlo desde el formulario.
- **usp_ActualizarProveedor:** Que permite la actualización de un proveedor identificado por su código. Mediante este procedimiento se puede activar o inactivar un proveedor, actualizando su estado, por lo que no será necesario recurrir a la eliminación de un proveedor.
- **usp_EliminarProveedor:** Que permite la eliminación física de un registro de proveedor. Aunque la aplicación no eliminara físicamente registro alguno, se recurrirá a este procedimiento en casos específicos.
- **usp_ConsultarProveedor:** Que permite obtener la información de un determinado proveedor mediante su código.
- **usp_ListarProveedor:** Que permite obtener un listado de todos los proveedores, desde la vista **vw_VistaProveedores**.

Todos estos procedimientos están ya en la base de datos **VentasLeon**.

Parte importante dentro del mantenimiento de proveedores es el desempeñado por la tabla **Tb_Ubigeo**. Se aprecia que en la tabla **Tb_Proveedor** se tiene al campo **Id_Ubigeo**, que es una FK que apunta a la PK **Id_Ubigeo** de la tabla **Tb_Ubigeo**. El **Id_Ubigeo** es un valor de 6 caracteres, donde los primeros 2 representan el departamento, el 3ro y 4to la provincia y el 5to y 6to el distrito. Por ejemplo, el distrito de Surquillo (que está en la provincia de Lima, departamento de Lima) su código de ubigeo sería este:



Para el manejo de estos elementos, se recurrirá a los procedimientos almacenados siguientes:

- **usp_Ubigeo_Departamentos:** Que genera una lista con el ID del departamento y su nombre.
- **usp_Ubigeo_ProvinciasDepartamento:** Que retorna una lista con el ID de las provincias y sus nombres de todas las provincias de un departamento dado.
- **usp_Ubigeo_DistritosProvinciaDepartamento:** Que retorna una lista con el ID de los distritos y sus nombres, de todos los distritos de una determinada provincia de un determinado departamento.

Estos procedimientos también están en la base de datos **VentasLeon**.

Con la explicación técnica del caso, procederemos a iniciar el mantenimiento de la tabla **Tb_Proveedor**.

Paso 1:

Abra la solución **DemoVentas3_Capas** y ubíquese en el proyecto **ProyVentas_BE** y copie a este proyecto la clase **ProveedorBE** entregada en su material. La clase es muestra los siguiente:

```
namespace ProyVentas_BE
{
    public class ProveedorBE
    {
        public String Cod_prv { get; set; }

        public String Raz_soc_prv { get; set; }

        public String Dir_prv { get; set; }

        public String Tel_prv { get; set; }

        public String Ruc_prv { get; set; }

        public String Rep_ven { get; set; }

        public String Id_Ubigeo { get; set; }

        public DateTime Fec_Registro { get; set; }

        public String Usu_Registro { get; set; }

        public DateTime Fec_Ult_Mod { get; set; }

        public String Usu_Ult_Mod { get; set; }

        public Int16 Est_prv { get; set; }
    }
}
```

Esta clase contiene todas las propiedades del proveedor que serán registradas en la tabla **Tb_Proveedor**, es decir, los campos de la tabla. Las entidades de negocio por tanto son clases que manejan la estructura de datos de la tabla a la que haremos mantenimiento.

Paso 2:

Ubíquese ahora en el proyecto **ProyVentas_ADO** que es la capa de datos. En este proyecto copie la plantilla de la clase **ProveedorADO** entregada en su material. En esta clase implementaremos los 5 métodos del mantenimiento de la tabla Tb_Proveedor. Recuerde que se deben establecer las líneas “using” siguientes:

```
using System.Data;
using System.Data.SqlClient;
using ProyVentas_BE;
```

Esto se hace con la intención de poder luego referenciar a la clase **DataSet** (dentro de **System.Data**), las clases de conexión a SQL Server (**System.Data.SqlClient**) y a la clase **ProveedorBE** (**ProyVentas_BE**).

También se han declarado todos los “insumos” para implementar el trabajo:

```
// Insumos.....
ConexionADO MiConexion = new ConexionADO();
SqlConnection cnx = new SqlConnection();
SqlCommand cmd = new SqlCommand();
SqlDataReader dtr;
```

No olvide que la clase **ConexionADO** (instancia **MiConexion**) con el método **GetCnx** nos permitirá obtener la cadena de conexión **Ventas** creada en el **app.Config** de la capa de presentación, proyecto **ProyVentas_GUI**, como ya se explicó en el mantenimiento de productos.

Procederemos a completar la codificación de los métodos de mantenimiento, tal como se aprecia en los siguientes recuadros:

```
public Boolean InsertarProveedor(ProveedorBE objProveedorBE)
{
    try
    {
        cnx.ConnectionString = MiConexion.GetCnx();
        cmd.Connection = cnx;
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.CommandText = "usp_InsertarProveedor";
        cmd.Parameters.Clear();
        // Pasamos los parametros al SP desde las propiedades de la entidad de negocios
        cmd.Parameters.AddWithValue("@vrz", objProveedorBE.Raz_soc_prv);
        cmd.Parameters.AddWithValue("@vdir", objProveedorBE.Dir_prv);
        cmd.Parameters.AddWithValue("@vtel", objProveedorBE.Tel_prv);
        cmd.Parameters.AddWithValue("@vruc", objProveedorBE.Ruc_prv);
        cmd.Parameters.AddWithValue("@vrep", objProveedorBE.Rep_ven);
        cmd.Parameters.AddWithValue("@vld_Ubigeo", objProveedorBE.Id_Ubigeo);
        cmd.Parameters.AddWithValue("@vUsu_Registro", objProveedorBE.Usu_Registro);
        cmd.Parameters.AddWithValue("@vEst_prv", objProveedorBE.Est_prv);

        // Ejecutamos....
        cnx.Open();
        cmd.ExecuteNonQuery();
        return true;
    }
    catch (SqlException x)
    {
        throw new Exception(x.Message);
        return false;
    }
    finally
    {
        if (cnx.State == ConnectionState.Open)
        {
            cnx.Close();
        }
    }
}
```

```
public Boolean ActualizarProveedor(ProveedorBE objProveedorBE)
{
    try
    {
        cnx.ConnectionString = MiConexion.GetCnx();
        cmd.Connection = cnx;
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.CommandText = "usp_ActualizarProveedor";
        //Agregamos parametros
        cmd.Parameters.Clear();
        // Pasamos los parametros al SP desde las propiedades de la entidad de negocios
        cmd.Parameters.AddWithValue("@vcod", objProveedorBE.Cod_prv );
        cmd.Parameters.AddWithValue("@vrz", objProveedorBE.Raz_soc_prv);
        cmd.Parameters.AddWithValue("@vdir", objProveedorBE.Dir_prv);
        cmd.Parameters.AddWithValue("@vtel", objProveedorBE.Tel_prv);
        cmd.Parameters.AddWithValue("@vruc", objProveedorBE.Ruc_prv);
        cmd.Parameters.AddWithValue("@vrep", objProveedorBE.Rep_ven);
        cmd.Parameters.AddWithValue("@vIdUbigeo", objProveedorBE.Id_Ubigeo);
        cmd.Parameters.AddWithValue("@vUsu_Ult_Mod", objProveedorBE.Usu_Ult_Mod);
        cmd.Parameters.AddWithValue("@vEst_prv", objProveedorBE.Est_prv);

        // Ejecutamos....
        cnx.Open();
        cmd.ExecuteNonQuery();
        return true;
    }
    catch (SqlException x)
    {
        throw new Exception(x.Message);
        return false;
    }
    finally
    {
        if (cnx.State == ConnectionState.Open)
        {
            cnx.Close();
        }
    }
}

public Boolean EliminarProveedor(String strcod)
{
    try
    {
        cnx.ConnectionString = MiConexion.GetCnx();
        cmd.Connection = cnx;
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.CommandText = "usp_EliminarProveedor";
        //Agregamos parametros
        cmd.Parameters.Clear();
        // Para eliminar un proveedor solo se requiere saber el codigo
        cmd.Parameters.AddWithValue("@vcod", strcod);

        cnx.Open();
        cmd.ExecuteNonQuery();
        return true;
    }
    catch (SqlException x)
    {
        throw new Exception(x.Message);
        return false;
    }
    finally
    {
        if (cnx.State == ConnectionState.Open)
        {
            cnx.Close();
        }
    }
}
```

```
public ProveedorBE ConsultarProveedor(String strCod)
{
    try
    {
        ProveedorBE objProveedorBE = new ProveedorBE();
        cnx.ConnectionString = MiConexion.GetCnx();
        cmd.Connection = cnx;
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.CommandText = "usp_ConsultarProveedor";
        cmd.Parameters.Clear();
        // Para la consulta de un proveedor solo se requiere el código del proveedor
        cmd.Parameters.AddWithValue("@vcod", strCod);
        cnx.Open();
        dtr = cmd.ExecuteReader();
        if (dtr.HasRows == true)
        {
            dtr.Read();
            // Asignamos las columnas del dtr a las propiedades de la instancia
            // con solo los datos que deseamos mostrar en la consulta
            objProveedorBE.Cod_prv = dtr["Cod_prv"].ToString();
            objProveedorBE.Raz_soc_prv = dtr["Raz_soc_prv"].ToString();
            objProveedorBE.Ruc_prv = dtr["Ruc_prv"].ToString();
            objProveedorBE.Dir_prv = dtr["Dir_prv"].ToString();
            objProveedorBE.Tel_prv = dtr["Tel_prv"].ToString();
            objProveedorBE.Rep_ven = dtr["Rep_ven"].ToString();
            objProveedorBE.Id_Ubigeo = dtr["Id_Ubigeo"].ToString();
            objProveedorBE.Est_prv = Convert.ToInt16(dtr["Est_prv"]);
            // Cerramos el dtr y devolvemos la instancia de la entidad de negocios
            dtr.Close();
        }
        return objProveedorBE;
    }
    catch (SqlException ex)
    {
        throw new Exception(ex.Message);
    }
    finally
    {
        if (cnx.State == ConnectionState.Open)
        {
            cnx.Close();
        }
    }
}
```

```
public DataTable ListarProveedor()
{
    try
    {
        DataSet dts = new DataSet();
        cnx.ConnectionString = MiConexion.GetCnx();
        cmd.Connection = cnx;
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.CommandText = "usp_ListarProveedor";
        cmd.Parameters.Clear();
        SqlDataAdapter ada = new SqlDataAdapter(cmd);
        ada.Fill(dts, "Proveedores");
        return dts.Tables["Proveedores"];
    }
    catch (SqlException ex)
    {
        throw new Exception(ex.Message);
    }
}
```

Como se aprecia, los métodos Insertar y Actualizar manejan como parámetro una instancia de la entidad de negocio **ProveedorBE**, donde se almacenan en sus propiedades los datos del nuevo proveedor o del proveedor actualizar. Ambos retornan un booleano, verdadero si la operación se efectúa sin problemas y falso si hay una excepción.

Para el caso del método Eliminar proveedor (que como se sabe es muy poco empleado, pero se incluye como parte del proceso) solo se requiere como parámetro el código del proveedor a eliminar y al igual que los 2 anteriores retorna un booleano indicando el éxito o fracaso de la operación. Estas 3 operaciones ejecutan el comando con el método **ExecuteNonQuery** como método de ejecución dado que los 3 realizan una tarea de actualización sobre la tabla **Tb_Proveedor**.

El método Consultar devuelve una instancia de la entidad de negocios **ProveedorBE**, en cuyas propiedades se tiene la información del proveedor consultado, manejando como parámetro el código de proveedor.

Por último, el método **Listar** devuelve un **DataTable** con la información de la ejecución del procedimiento almacenado **usp_ListarProveedor**, empleando mecanismos del entorno desconectado, tal como se explicó en clases.

En cuanto al manejo del ubigeo, en la plantilla se entregó la clase **UbigeoADO**, la cual debe copiar en el proyecto **ProyVentas_ADO** y que tiene 3 métodos, mostrados a continuación:

```
public DataTable Ubigeo_Departamentos()
{
    DataSet dts = new DataSet();
    try
    {
        cnx.ConnectionString = MiConexion.GetCnx();
        cmd.Connection = cnx;
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.CommandText = "usp_Ubigeo_Departamentos";
        cmd.Parameters.Clear();
        SqlDataAdapter miada;
        miada = new SqlDataAdapter(cmd);
        miada.Fill(dts, "Departamentos");
        return dts.Tables["Departamentos"];
    }
    catch (SqlException ex)
    {
        throw new Exception(ex.Message);
    }
}
```

```
public DataTable Ubigeo_ProvinciasDepartamento(String strIdDepartamento)
{
    DataSet dts = new DataSet();
    try
    {
        cnx.ConnectionString = MiConexion.GetCnx();
        cmd.Connection = cnx;
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.CommandText = "usp_Ubigeo_ProvinciasDepartamento";
        cmd.Parameters.Clear();
        cmd.Parameters.AddWithValue("@IdDepartamento", strIdDepartamento);
        SqlDataAdapter miada;
        miada = new SqlDataAdapter(cmd);
        miada.Fill(dts, "Provincias");
        return dts.Tables["Provincias"];
    }
    catch (SQLException ex)
    {
        throw new Exception(ex.Message);
    }
}

public DataTable Ubigeo_DistritosProvinciaDepartamento(String strIdDepartamento, String strIdProvincia)
{
    DataSet dts = new DataSet();
    try
    {
        cnx.ConnectionString = MiConexion.GetCnx();
        cmd.Connection = cnx;
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.CommandText = "usp_Ubigeo_DistritosProvinciaDepartamento";
        cmd.Parameters.Clear();
        cmd.Parameters.AddWithValue("@IdDepartamento", strIdDepartamento);
        cmd.Parameters.AddWithValue("@IdProvincia", strIdProvincia);
        SqlDataAdapter miada;
        miada = new SqlDataAdapter(cmd);
        miada.Fill(dts, "Distritos");
        return dts.Tables["Distritos"];
    }
    catch (SQLException ex)
    {
        throw new Exception(ex.Message);
    }
}
```

Como se aprecia estos métodos devuelven instancias de la clase DataTable con todos los departamentos (método **Ubigeo_Departamentos**), las provincias de un departamento (método **Ubigeo_ProvinciasDepartamento**), y los distritos de una provincia de un departamento dado (método **Ubigeo_DistritosProvinciaDepartamento**). Se emplean los procedimientos almacenados vinculados al tema del Ubigeo citados líneas arriba.

Paso 3:

Terminada la codificación en la capa de datos, pasaremos ahora al proyecto **ProyVentas_BL**. Recordar que esta capa es un nexo entre la capa de datos y la capa de presentación. Copie a este proyecto las clases **ProveedorBL** y **UbigeoBL** entregados en su plantilla. Como se observa en la clase **ProveedorBL**, se tienen los “using” siguientes:

```
using System.Data;
using ProyVentas_ADO;
using ProyVentas_BE;
```


Dado que se va a emplear una instancia de la clase **DataTable** (System.Data), una instancia de la clase **ProveedorADO** (ProyVentas_ADO) y una instancia de la clase **ProveedorBE** (ProyVentas_BE), por tanto, estos using son relevantes.

La clase **ProveedorBL** contiene la invocación a los métodos de la capa de datos por medio de la instancia **objProveedorADO** de la clase **ProveedorADO** como se muestra en la imagen:

```
public class ProveedorBL
{
    // Creamos una instancia de la clase ProveedorADO para invocar a sus metodos.
    ProveedorADO objProveedorADO = new ProveedorADO();

    public Boolean InsertarProveedor(ProveedorBE objProveedorBE)
    {
        return objProveedorADO.InsertarProveedor(objProveedorBE);
    }
    public Boolean ActualizarProveedor(ProveedorBE objProveedorBE)
    {
        return objProveedorADO.ActualizarProveedor(objProveedorBE);
    }
    public Boolean EliminarProveedor(String strcod)
    {
        return objProveedorADO.EliminarProveedor(strcod);
    }

    public ProveedorBE ConsultarProveedor(String strcod)
    {
        return objProveedorADO.ConsultarProveedor(strcod);
    }

    public DataTable ListarProveedor()
    {
        return objProveedorADO.ListarProveedor();
    }
}
```

De manera similar la clase **UbigeoBL**, establece un llamado a los métodos de la clase **UbigeoADO**:

```
using System.Threading.Tasks;
// Agregamos los using...
using ProyVentas_ADO;
using ProyVentas_BE;
using System.Data;
namespace ProyVentas_BL
{
    public class UbigeoBL
    {
        UbigeoADO objUbigeoADO = new UbigeoADO();

        public DataTable Ubigeo_Departamentos()
        {
            return objUbigeoADO.Ubigeo_Departamentos();
        }
        public DataTable Ubigeo_ProvinciasDepartamento(String strIdDepartamento)
        {
            return objUbigeoADO.Ubigeo_ProvinciasDepartamento(strIdDepartamento);
        }
        public DataTable Ubigeo_DistritosProvinciaDepartamento(String strIdDepartamento, String strIdProvincia)
        {
            return objUbigeoADO.Ubigeo_DistritosProvinciaDepartamento(strIdDepartamento, strIdProvincia);
        }
    }
}
```

Estas clases ya están íntegramente codificadas desde su planilla. Por lo tanto y para verificar que todo lo avanzado hasta aquí este correcto, compile su solución. De existir un error por favor corregirlo y vuelva a compilar.

Pasaremos ahora a implementar la capa de presentación en el proyecto **ProyVentas_GUI**.

Paso 4:

Ubíquese en la capa de presentación, es decir, proyecto **ProyVentas_GUI**. Trabajaremos el mantenimiento con los formularios **ProveedorMan01**, para mostrar el listado general de proveedores y desde ahí, invocar a los formularios **ProveedorMan02** para la inserción de un nuevo proveedor y **ProveedorMan03** para la actualización de un proveedor.

NOTA: Estos formularios deben ser copiados al proyecto **ProyVentas_GUI** desde los materiales entregados en la sesión.

Seleccione el formulario **ProveedorMan01** para completar su codificación. Desde la ventana de código se aprecia que en la zona de “using” se han definido el using para **ProyVentas_BL** dado que se empleara una instancia de la clase **ProveedorBL**. Recuerda que la capa de presentación no “conversa” directamente con la capa de datos, sino que lo hace con la capa de negocios como intermediaria, garantizando una independencia entre la capa de datos y la de presentación.

En la zona declaraciones se tiene lo siguiente:

```
ProveedorBL objProveedorBL = new ProveedorBL();  
DataView dtv;
```

Se tiene una instancia de la clase **ProveedorBL** (**objProveedorBL**) como ya se anticipó y una instancia de la clase **DataView** (**dtv**). La clase **DataView** es parte del entorno desconectado y nos permitirá filtrar en memoria los proveedores por medio de su razón social sin emplear conexión como se verá en breve. La codificación se inicia con el método **CargarDatos**:

```
public void CargarDatos(String strFiltro)  
{  
    // Construimos el objeto DataView dtv en base al DataTable devuelto por el metodo ListarProveedor  
    // Y lo filtramos de acuerdo al parametro strFiltro  
    dtv = new DataView(objProveedorBL.ListarProveedor());  
    dtv.RowFilter = "raz_soc_prv like '%" + strFiltro + "%'";  
    dtgProveedor.DataSource = dtv;  
    lblRegistros.Text = dtgProveedor.Rows.Count.ToString();  
}
```

Este método permite enlazar el datagrid **dtgProveedor** al objeto **dtv** previamente filtrado (propiedad **RowFilter**) con la expresión “raz_soc_prv like '%" + strFiltro + "%'” de tal manera que solo se filtren los proveedores que tengan dentro de su razón social la cadena almacenada en la variable **strFiltro**, pasada como parámetro del método. La ventaja de la instancia **dtv** (de la clase

DataGridView) es que hace el filtro en memoria, sin emplear recurso de conexión. Además, el método muestra en el label **lblRegistros** la cantidad de filas filtradas en el datagrid **dtgProveedor**.

Completaremos la codificación del evento Load del formulario:

```
private void ProveedorMan01_Load(object sender, EventArgs e)
{
    try
    {
        // Invocamos al metodo CargarDatos pasandole una cadena vacia ,
        // lo cual hara que se muestren todos los proveedores por defecto al momento de cargar el formulario
        //
        CargarDatos("");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error:" + ex.Message);
    }
}
```

El comentario incluido nos exonera de mayor explicación.

Ahora codificaremos el evento **TextChanged** de la caja de texto **txtFiltro**. La idea es que el filtro de proveedores se aplique a medida que el usuario va digitando caracteres en esta caja de texto, con la ventaja que el filtro se dará en memoria sin consumir conexión como ya se indicó.

```
private void txtFiltro_TextChanged(object sender, EventArgs e)
{
    try
    {
        // Pasaremos al metodo CargarDatos el texto que se va escribiendo
        // en la caja de texto
        CargarDatos(txtFiltro.Text.Trim());
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error:" + ex.Message);
    }
}
```

Asegúrate que el proyecto de inicio sea el **ProyVentas_GUI** (dándole clic derecho desde la ventana del Explorador de Soluciones y eliges la opción “Establecer como proyecto de inicio”). Luego, en la clase **Program** indica que el formulario de inicio será **ProveedorMan01** y ejecuta.

En primera instancia se muestra el formulario listando todos los campos devueltos por el procedimiento almacenado **usp_ListarProveedor** invocado desde la capa de datos por el método **ListarProveedor** de la capa de datos y que a su vez es llamado desde la capa de negocios. Se verá esto:

Mantenimiento de Proveedores

Ingrese iniciales de razon social:

Cod_prv	Raz_soc_prv	Dir_prv	Tel_prv	Ruc_Priv	Rep_ven	Id_Ubigeo	Departament	Provincia	Distrito	Fec_Registro	Usu_Registro	Fec_Ult_Mod	Usu_Ult_Mod	Est_prv	Estado
V003	3M	Av. Venezu...	2908165	20559649636	Omar Isola	140142	LIMA	LIMA	LOS OLIVOS	1/01/2020	Jeon			0	Inactivo
V137	ACABADOS...	CALLE JUL...		20601706351	Wendy Mac...	040211	AREQUIPA	CAYLLOMA	LLUTA	10/09/2020...	Jeon			1	Activo
V005	Acker	Calle Las A...	4780143	20601242410	Julio Zamora	141106	LIMA	CALLAO	VENTANILLA	1/01/2020	Jeon	5/02/2021...	User1	1	Activo
V099	ADONAI	AV MARIA ...	2658503	20561166138	Liz Nixon ...	140106	LIMA	LIMA	COMAS	10/09/2020...	Jeon	21/02/2021...	Jeon	0	Inactivo
V172	ALBERTO ...	Av. El camp...		10036604650		040202	AREQUIPA	CAYLLOMA	ACHOMA	10/09/2020...	Jeon			1	Activo
V192	ALBRESA ...			10040575648	-	040128	AREQUIPA	AREQUIPA	ALTO SELV...	10/09/2020...	Jeon			1	Activo
V163	AMERICAN...	JR. ANTONI...	5332069	10293869168	-	040212	AREQUIPA	CAYLLOMA	MACA	10/09/2020...	Jeon			1	Activo
V189	ANDRES Q...	Av. ISIL 1730	27	10060075382	Juan Diaz	040304	AREQUIPA	CAMANA	MARISCAL ...	10/09/2020...	Jeon			1	Activo
V191	ANIBAL DO...			10412593940	-	040212	AREQUIPA	CAYLLOMA	MACA	10/09/2020...	Jeon			1	Activo
V023	AROCA	Urb. LOS R...	-	20543835685	Maria Ander...	140104	LIMA	LIMA	BREÑA	10/09/2020...	Jeon	15/02/2021...	Jeon	1	Activo
V114	ARONES H...	CALLE LOS...		20600375360	Charlotte Co...	140125	LIMA	LIMA	BARRANCO	10/09/2020...	Jeon			1	Activo
V050	ASPEN INV...	JR. PEDRO...	3561072	20534639245	Lino Rodrig...	140139	LIMA	LIMA	CIENEGUIL...	10/09/2020...	Jeon			1	Activo
V002	Atlas	Av. Lima 471	9874560	20600587529	Cesar Torres	140133	LIMA	LIMA	JESUS MA...	1/01/2020	Jeon			1	Activo

Registros: 206

Insertar Actualizar Salir

Si escribe el texto "alb" en la caja de texto txtFiltro, a medida que escribe se filtran los proveedores que contienen lo escrito dentro de la razón social, quedando así al final:

Mantenimiento de Proveedores

Ingrese iniciales de razon social:

Cod_prv	Raz_soc_prv	Dir_prv	Tel_prv	Ruc_Priv	Rep_ven	Id_Ubigeo	Departamento	Provincia	Distrito	Fec_Registro	Usu_Registro	Fec_Ult_Mod	Usu_Ult_Mod	Est_prv	Estado
V172	ALBERTO ...	Av. El camp...		10036604650		040202	AREQUIPA	CAYLLOMA	ACHOMA	10/09/2020...	Jeon			1	Activo
V192	ALBRESA A...			10040575648	-	040128	AREQUIPA	AREQUIPA	ALTO SELV...	10/09/2020...	Jeon			1	Activo
V102	LUIS ALBE...	ABANCAY 4...	4282265	20454516746	Miguel Ange...	140101	LIMA	LIMA	LIMA	10/09/2020...	Jeon			1	Activo

Registros: 3

Insertar Actualizar Salir

Si desea mostrar todos los proveedores de nuevo, limpie la caja de texto. ¡ Súper fácil!!!!

Nota: Mas adelante explicaremos como formatear el DataGridView para que no se muestren todas las columnas, sino solo las más importantes.

Paso 5:

Iniciaremos ahora el camino para la inserción de nuevos proveedores, de la mano del formulario **ProveedorMan02**. Este formulario deberá ser lanzado de forma modal desde el formulario **ProveedorMan01**, al hacer clic en el botón **btnInsertar**. Completamos el código de este evento:

```
private void btnInsertar_Click(object sender, EventArgs e)
{
    try
    {
        // Creamos una instancia de ProveedorMan02 y lo mostramos de manera modal
        ProveedorMan02 prov02 = new ProveedorMan02();
        prov02.ShowDialog();

        // Al retornar, refrescamos la vista y cargamos los datos para visualizar
        // al proveedor agregado
        dtv = new DataView(objProveedorBL.ListarProveedor());
        CargarDatos(txtFiltro .Text .Trim ());
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error : " + ex.Message);
    }
}
```

Seleccionemos el formulario **ProveedorMan02** y vayamos a la ventana de código a revisar lo que la plantilla tiene. Observamos en la zona de “using” lo siguiente:

```
using System.Text;
using System.Windows.Forms;
// Agregamos
using ProyVentas_BL;
using ProyVentas_BE;
namespace ProyVentas_GUI
{
    public partial class ProveedorMan02 : Form
    {
        // Declaramos las instancias...
        ProveedorBL objProveedorBL = new ProveedorBL();
        ProveedorBE objProveedorBE = new ProveedorBE();
    }
}
```

Vemos la declarativa a usar de los proyectos **ProyVentas_BE** y **ProyVentas_BL**, dado que emplearemos instancias de las clases **ProveedorBE**, en la que asignaremos las propiedades (datos) del nuevo proveedor y de **ProveedorBL**, que es la que posee el método **InsertarProveedor**, la cual a su vez llamara al mismo método en la capa de datos. Las instancias son **objProveedorBE** y **objProveedorBL**, tal como se aprecia. Además, será necesaria la declaración de una instancia de la clase **UbigeoBL**.

Al momento de lanzar el formulario **ProveedorMan02** debemos cargar los 3 combobox que permitirán seleccionar el departamento, provincia y distrito. Por ello en el evento Load debemos codificar lo siguiente:

```
private void ProveedorMan02_Load(object sender, EventArgs e)
{
    try
    {
        // Cargamos los combos de Ubigeo . Por defecto elegiremos Lima, Lima , Lima (14,01,01)
        CargarUbigeo("14","01","01");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error : " + ex.Message);
    }
}
```

Al cargar el formulario invocamos a un método llamado **CargarUbigeo** y le pasamos el Id del ubigeo del departamento, provincia y distrito de Lima "14","01","01". El método se describe a continuación para codificarlo en la plantilla:

```
private void CargarUbigeo(String IdDeps,String IdProv,String IdDist)
{
    UbigeoBL objUbigeoBL = new UbigeoBL();
    cboDepartamento.DataSource = objUbigeoBL.Ubigeo_Departamentos();
    cboDepartamento.ValueMember = "IdDeps";
    cboDepartamento.DisplayMember = "Departamento";
    cboDepartamento.SelectedValue = IdDeps ;

    cboProvincia .DataSource = objUbigeoBL.Ubigeo_ProvinciasDepartamento (IdDeps);
    cboProvincia.ValueMember = "IdProv";
    cboProvincia.DisplayMember = "Provincia";
    cboProvincia.SelectedValue = IdProv ;

    cboDistrito.DataSource = objUbigeoBL.Ubigeo_DistritosProvinciaDepartamento (IdDeps ,IdProv);
    cboDistrito.ValueMember = "IdDist";
    cboDistrito.DisplayMember = "Distrito";
    cboDistrito.SelectedValue =IdDist;

}
```

Declaramos una instancia de la clase UbigeoBL llamada **objUbigeoBL** y con ella invocamos primero al método **Ubigeo_Departamento** que retorna un **DataTable** con los Id y nombre de departamentos, la cual se enlaza al combobox **cboDepartamento**, manejando como columna de valor el campo **IdDeps** (Id departamento) y como campo de despliegue el campo **Departamento**, es decir el nombre del departamento. Se selecciona el valor del parámetro **IdDeps** ("14") que es el parámetro pasado al método.

Para el tema de las provincias se hace algo similar. Se invoca al método **Ubigeo_Provincias_Departamento**, el cual maneja como argumento el parámetro **Idea** pasado al método ("14"). Con ello devolverá un **Datatable** con todas las provincias del departamento de "14", o sea Lima. Se enlaza este **DataTable** al combo **cboProvincia**, manejando como campo de valor el **IdProv** y como campo de visualización el campo **Provincia**.

El caso se repite para llenar el combo **cboDistritos**, invocando al método **Ubigeo_Distrito_Provincia_Departamento**, pasándole los parámetros recibidos en el método **IdDepa** y **IdProv** ("14" y "01" respectivamente) y se enlaza dicho combo al **DataTable** que devuelve todos los distritos de la provincia y del departamento pasados como parámetros. Se maneja como campo de valor el **IdDist** y como campo de visualización el **Distrito**.

Ahora, es necesario que al seleccionar un departamento los combos de provincia y distrito se actualicen "en cascada", así como al seleccionar una provincia el combo de distritos actualice sus valores. Esto es algo muy conocido en las aplicaciones que manejan el concepto de ubigeo. Para implementar este efecto "cascada" la plantilla ya tiene codificados los eventos **SelectionChangeComitted** de los combos de departamento y provincia, tal como se muestra a continuación:

```
private void cboDepartamento_SelectionChangeComitted(object sender, EventArgs e)
{
    // Refrescamos los combos con la primera provincia y el primer distrito de esa provincia para el departamento
    // seleccionado del combo cboDepartamento
    CargarUbigeo(cboDepartamento.SelectedValue.ToString(), "01", "01");
}

private void cboProvincia_SelectionChangeComitted(object sender, EventArgs e)
{
    // Refrescamos los combos con el primer distrito para el departamento y provincia
    // seleccionado del combo cboDepartamento y cboProvincia
    CargarUbigeo(cboDepartamento.SelectedValue.ToString(), cboProvincia.SelectedValue.ToString(), "01");
}
```

El evento **SelectionChangeComitted** es el que se debe emplear cuando los combos son llenados desde **DataTables** y se requiere del efecto "cascada" como se mencionó.

Por último completaremos la codificación del botón **btnGrabar** para la inserción del nuevo proveedor. Veamos el código:

```
private void btnGrabar_Click(object sender, EventArgs e)
{
    try
    {
        // Validar que este con valor la razon social
        if (txtRS.Text.Trim() == String.Empty)
        {
            throw new Exception("La razon social es obligatoria.");
        }
        // Validamos que el RUC este lleno
        if (mskRuc.MaskFull == false)
        {
            throw new Exception("El RUC debe tener 11 caracteres.");
        }
        // Pasamos los valores a las propiedades de la instancia...
        objProveedorBE.Raz_soc_prv = txtRS.Text.Trim();
        objProveedorBE.Dir_prv = txtDir.Text.Trim();
        objProveedorBE.Tel_prv = txtTel.Text.Trim();
        objProveedorBE.Ruc_prv = mskRuc.Text.Trim();
        objProveedorBE.Rep_ven = txtRepVen.Text.Trim();
        // Recuerde que el IdUbigeo es la concatenacion de los valores del Id Departamento,
        // Id Provincia y Id Distrito seleccionados desde los respectivos combos
        objProveedorBE.Id_Ubigeo = cboDepartamento.SelectedValue.ToString() + cboProvincia.SelectedValue.ToString() +
            cboDistrito.SelectedValue.ToString();
        // Se asignara por ahora al usuario jleon como usuario de registro
        objProveedorBE.Usu_Registro = "jleon";
        objProveedorBE.Est_prv = Convert.ToInt16 (chkEstado.Checked);
        if (objProveedorBL.InsertarProveedor(objProveedorBE) == true)
        {
            this.Close();
        }
        else
        {
            throw new Exception("No se inserto el registro. Contacte con IT.");
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Se ha producido el error: " + ex.Message);
    }
}
```

Se observa que se valida la obligatoriedad del ingreso valores para la razón social y el llenado de la máscara del RUC. Si las validaciones se cumplen se procede a asignar a las propiedades de la instancia **objProveedorBE** los valores ingresados desde los controles del formulario. Vea como se concatenan las propiedades **SelectedValue** de los 3 combos del ubigeo y se asigna a la propiedad **Id_Ubigeo** del nuevo proveedor. Por ahora, al no tener implementada la parte del formulario Login, se asignará como usuario de registro a "jleon".

Nota: Si ya implemento el formulario Login con su instructor, reemplace el "jleon" por el usuario logueado, según lo explicado.

Asignadas las propiedades se invoca al método **InsertarProveedor** de la instancia **objProveedorBL**, pasándole como argumento la instancia **objProveedorBE** con toda la información del nuevo proveedor. Si el método retorna verdadero quiere decir que todo salió bien y se cierra el formulario, volviendo el control al formulario **ProveedorMan01**, de lo contrario se provoca una excepción indicando el problema. Ejecute el proyecto desde el formulario **ProveedorMan01** e inserte 2 nuevos proveedores y verifique en la base de datos su existencia, así como su aparición en el **DataGridView** del formulario **ProveedorMan01**.

Paso 6:

Ahora manejaremos lo referente a la actualización de un proveedor. Codificaremos el evento click del botón **btnActualizar** del formulario **ProveedorMan01**, tal como se aprecia:

```
private void btnActualizar_Click(object sender, EventArgs e)
{
    try
    {
        ProveedorMan03 prov03 = new ProveedorMan03();
        // Se toma el valor de la columna cero de la fila seleccionada en el
        // datagridview ....
        prov03.Codigo = dtgProveedor.CurrentRow.Cells[0].Value.ToString();
        prov03.ShowDialog();

        // Al retornar, refrescamos la vista y cargamos los datos para ver los
        // cambios del proveedor actualizado
        dtv = new DataView(objProveedorBL.ListarProveedor());
        CargarDatos(txtFiltro .Text .Trim());
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error : " + ex.Message);
    }
}
```

Con esta rutina se asigna a la propiedad Codigo del formulario **ProveedorMan03** (instancia prov03), el código del proveedor que se desea actualizar tomándolo desde la celda 0 de la fila seleccionada en el **Datagrid dtgProveedor** y luego se carga de manera modal la instancia **provo03**.

Paso 7:

Nos colocamos en el formulario **ProveedorMan03** y vemos en la ventana de código que ya se tienen las instancias de **ProveedorBL** y **ProveedorBE** en la zona de declaraciones, así como la definición de la propiedad **Codigo**, invocada desde el evento click del botón **btnActualizar** del formulario **ProveedorMan01**:

```
ProveedorBL objProveedorBL = new ProveedorBL();
ProveedorBE objProveedorBE = new ProveedorBE();

public ProveedorMan03()
{
    InitializeComponent();
}

// Creamos la propiedad Codigo ,que recepcionara el valor del codigo del proveedor
// a actualizar enviado desde el formulario ProveedorMan01

public String Codigo { get; set; }

}
```

Para que al momento de cargar este formulario se muestren todos los datos del proveedor a actualizar, codifiquemos el evento **Load** tal como se indica:

```
private void ProveedorMan03_Load(object sender, EventArgs e)
{
    try
    {
        // Mostramos los datos del proveedor a actualizar
        objProveedorBE = objProveedorBL.ConsultarProveedor(this.Codigo);

        lblCod.Text = objProveedorBE.Cod_prv;
        txtRS.Text = objProveedorBE.Raz_soc_prv;
        txtDir.Text = objProveedorBE.Dir_prv;
        txtTel.Text = objProveedorBE.Tel_prv;
        mskRuc.Text = objProveedorBE.Ruc_prv;
        txtRepVen.Text = objProveedorBE.Rep_ven;
        chkEstado.Checked = Convert.ToBoolean(objProveedorBE.Est_prv);
        String Id_Ubigeo = objProveedorBE.Id_Ubigeo;
        // Mostramos en los 3 combos el ubigeo
        // Caracteres 1 y 2 : Departamento, Caracteres 3 y 4 : Provincia, Caracteres 5 y 6 : Distrito
        // Cargamos el Ubigeo
        CargarUbigeo(Id_Ubigeo.Substring(0, 2), Id_Ubigeo.Substring(2, 2), Id_Ubigeo.Substring(4, 2));
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error : " + ex.Message);
    }
}
```

Tal como se aprecia, se invoca al método **ConsultarProveedor**, pasándole como argumento la propiedad **Codigo** (asignada desde el **ProveedorMan01**). Los datos del proveedor se asignan a la instancia **objProveedorBE** y cada propiedad de esta instancia se muestra en los controles del formulario. Para el caso de los 3 combos que componen el ubigeo primero se asigna a una variable **Id_Ubigeo** la propiedad del mismo nombre de la instancia **objProveedorBE** y se invoca al método **CargarUbigeo** con los parámetros de departamento, provincia y distrito respectivos. El método **CargarUbigeo** contiene esto :

```
private void CargarUbigeo(String IdDepa, String IdProv, String IdDist)
{
    UbigeoBL objUbigeoBL = new UbigeoBL();
    cboDepartamento.DataSource = objUbigeoBL.Ubigeo_Departamentos();
    cboDepartamento.ValueMember = "IdDepa";
    cboDepartamento.DisplayMember = "Departamento";
    cboDepartamento.SelectedValue = IdDepa;

    cboProvincia.DataSource = objUbigeoBL.Ubigeo_ProvinciasDepartamento(IdDepa);
    cboProvincia.ValueMember = "IdProv";
    cboProvincia.DisplayMember = "Provincia";
    cboProvincia.SelectedValue = IdProv;

    cboDistrito.DataSource = objUbigeoBL.Ubigeo_DistritosProvinciaDepartamento(IdDepa, IdProv);
    cboDistrito.ValueMember = "IdDist";
    cboDistrito.DisplayMember = "Distrito";
    cboDistrito.SelectedValue = IdDist;
}
```

ISIL-Aplicaciones I

Paso 8:

Compile su proyecto y ejecute. Seleccione un proveedor desde el **DataGrid** y al hacer click en el botón **btnActualizar** se carga el formulario **ProveedorMan03** con los datos del proveedor seleccionado. Si por ejemplo selecciono desde el DataGrid al proveedor **V005** y hace click en el botón Actualizar aparece en forma modal el formulario **ProveedorMan03**.

Mantenimiento de Proveedores

Ingrese iniciales de razon social:

Codigo	Razon Social	Direccion	Telefono	Departament	Provincia	Distrito	Estado	Ruc_Priv	Rep_ven	Id_Ubigeo	Fec_Registro	Usu_Registro	Fec_Ult_Mod	Usu_Ult_Mod	Est_priv
V003	3M	Av. Venezu...	2908165	LIMA	LIMA	LOS OLIVOS	Inactivo	20559649636	Omar Isola	140142	1/01/2020	jleon			0
V137	ACABADO...	CALLE JULI...		AREQUIPA	CAYLLOMA	LLUTA	Activo	20601706351	Wendy Ma...	040211	10/09/2020...	jleon			1
V005	Acker	Calle Las A...	4780143	LIMA	CALLAO	VENTANILLA	Activo	20601242410	Julio Zamora	141106	1/01/2020	jleon	5/02/2021...	User1	1
V099	ADONAI	AV MARIA ...	2658503	LIMA	LIMA	COMAS	Inactivo	20561166138	Liz Nixon ...	140106	10/09/2020...	jleon	21/02/2021...	jleon	0
V172	ALBERTO O...	Av. El cam...		AREQUIPA	CAYLLOMA	ACHOMA	Activo	10036604650		040202	10/09/2020...	jleon			1
V192	ALBRESA A...			AREQUIPA	AREQUIPA	ALTO SELVA...	Activo	10040575648	-	040128	10/09/2020...	jleon			1
V163	AMERICA...	JR.ANTONI...	5332069	AREQUIPA	CAYLLOMA	MACA	Activo	10293869168	-	040212	10/09/2020...	jleon			1
V189	ANDRES Q...	Av. ISIL 1730	27	AREQUIPA	CAMANA	MARISCAL ...	Activo	10060075382	Juan Diaz	040304	10/09/2020...	jleon			1
V191	ANIBAL D...			AREQUIPA	CAYLLOMA	MACA	Activo	10412593940	-	040212	10/09/2020...	jleon			1
V023	AROCA	Urb. LOS R...	-	LIMA	LIMA	BREÑA	Activo	20543835685	Maria And...	140104	10/09/2020...	jleon	15/02/2021...	jleon	1
V114	ARONES H...	CALLE LOS...		LIMA	LIMA	BARRANCO	Activo	20600375360	Charlotte C...	140125	10/09/2020...	jleon			1
V050	ASPEN INV...	JR. PEDRO ...	3561072	LIMA	LIMA	CIENEGUIL...	Activo	20534639245	Lino Rodri...	140139	10/09/2020...	jleon			1
V002	Atlas	Av. Lima 471	9874560	LIMA	LIMA	JESUS MARIA	Activo	20600587529	Cesar Torres	140133	1/01/2020	jleon			1
V024	AZEMAR P...	SAN ENRIQ...	4587581	LIMA	LIMA	SANTIAGO ...	Activo	20602014925	Ana Trujillo...	140130	10/09/2020...	jleon			1

Registros: 206

Insertar Actualizar Salir

Actualizar Proveedor

Datos

Codigo: V005

R.Social: Acker

Direccion: Calle Las Amapolas 600

Telefono: 4780143

RUC: 20601242410

Departamento: LIMA

Provincia: CALLAO

Distrito: VENTANILLA

Rep. Ventas: Julio Zamora

Estado: ☒ Activo?

Grabar Cancelar

Fec_Ult_Mod Usu_Ult_Mod Est_priv

5/02/2021 ... User1 1

21/02/2021... jleon 0

15/02/2021... jleon 1

Registros: 206

Insertar Actualizar Salir

Concluya Ud. el mantenimiento de la tabla **Tb_Proveedor** codificando el evento click de botón **btnGrabar** del formulario **ProveedorMan03**, invocando al método **ActualizarProveedor** de la instancia **objProveedorBL**. ¡¡¡ A programar!!!