

Curso de Patrones y Buenas Prácticas .Net

Arquitectura de Aplicaciones .Net (Capas)

Practica

Plataforma: Visual Studio .Net
Framework: Microsoft .Net Framework 2.0

Documento de Referencia y Capacitación
(Este documento está sujeto a cambios)

Fecha de Creación: 03-Jun-2010
Versión: 1.1.0.0
Autor: Julio Cesar Robles Uribe

Tabla de Contenido

Introducción	4
Práctica.....	5
Solución.....	6
Entidades (Entities).....	6
TelephoneType	6
Telephone	9
PersonLight.....	13
Lógica de Acceso a Datos (DAL)	16
TelephoneDAL	16
PersonDAL	23
Lógica de Negocios (BL).....	24
TelephoneBL.....	24
PersonBL	27
Interfaz de Usuario (UI)	28
FrmTelephones	28
FrmPersons.....	46
MDIMain.....	49
Acerca de	54
Código MDIMain	60

Introducción

Este documento proporciona una breve una posible solución para la práctica del ejercicio de Arquitectura de aplicaciones.

Práctica

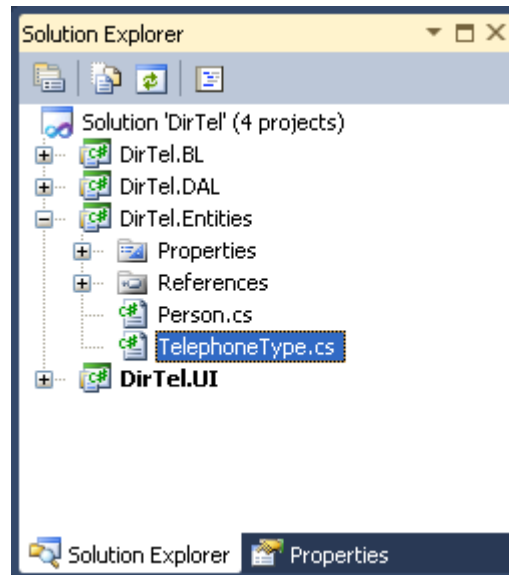
1. Implemente la funcionalidad para el manejo de teléfonos de las personas, tenga en cuenta que una persona puede tener muchos teléfonos (Casa, Celular, FAX, etc). Para ello cree sus propios Scripts de tablas y paquetes o procedimientos de base de datos.
2. Adicione un menú para hacer el llamado de cada uno de los formularios.

Solución

Entidades (Entities)

TelephoneType

1. Agregue una nueva clase Entidad denominada **TelephoneType** para manejar los datos de los tipos de los teléfonos.

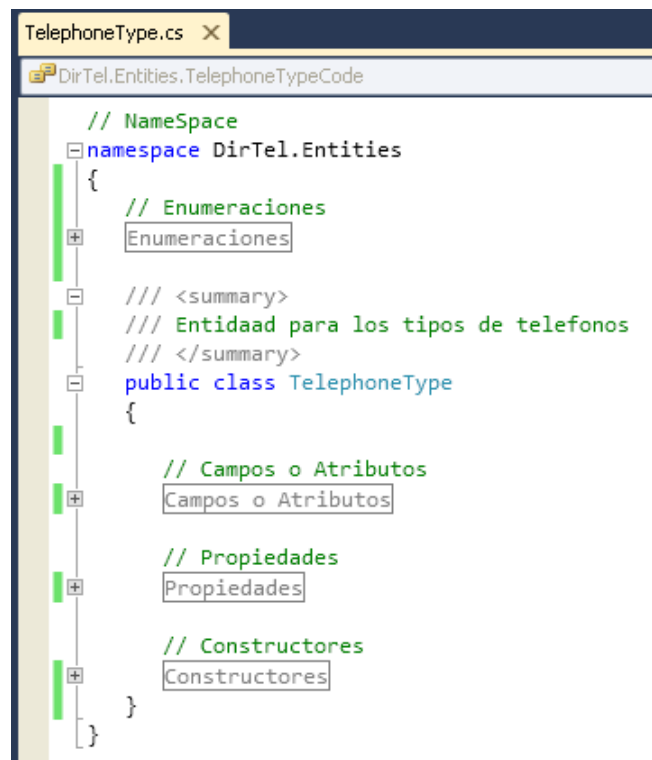


2. Edite el contenido del archivo **TelephoneType.cs** y adicione el encabezado de la documentación correspondiente.

```
TelephoneType.cs X
DirTel.Entities.TelephoneType

#region Documentación
/
* Propiedad intelectual de Engineers & Tools (c).
*****
* Unidad      : TelephoneType.cs
* Descripción : Entidad para manejar los tipos de telefonos
* Autor       : Julio Cesar Robles Uribe - Jucer
* Fecha       : 21-May-2010
*
* Fecha      Autor      Modificación
* =====
* 21-May-2010 Jucer      1 - Version Inicial
*****/
#endregion Documentación
```

3. Modifique el alcance de la clase a **public**, adicione la documentación y los comentarios correspondientes.



4. Adicione una enumeración denominada **TelephoneTypeCode** con la siguiente definición, por fuera de la clase

```
/// <summary>
/// Identificadores de los Tipos de Telefonos
/// </summary>
public enum TelephoneTypeCode
{
    /// <summary>
    /// Hogar o Fijo
    /// </summary>
    Home = 1,
    /// <summary>
    /// Celular o Movil
    /// </summary>
    Mobile,
    /// <summary>
    /// Oficina
    /// </summary>
    Work
}
```

5. Agregue los campos y atributos de la siguiente forma:

```
// Identificador delCodigo del tipo de telefono
private TelephoneTypeCode value;
// Descripcion onombre del tipo de telefono
private string name;
```

6. Cree las propiedades así:

```
/// <summary>
/// Identificador del Código del tipo de telefono
/// </summary>
public TelephoneTypeCode Value
{
    get
    {
        return this.value;
    }
    set
    {
        this.value = value;
    }
}

/// <summary>
/// Descripción o nombre del tipo de telefono
/// </summary>
public string Name
{
    get
    {
        return this.name;
    }
    set
    {
        this.name = value;
    }
}
```

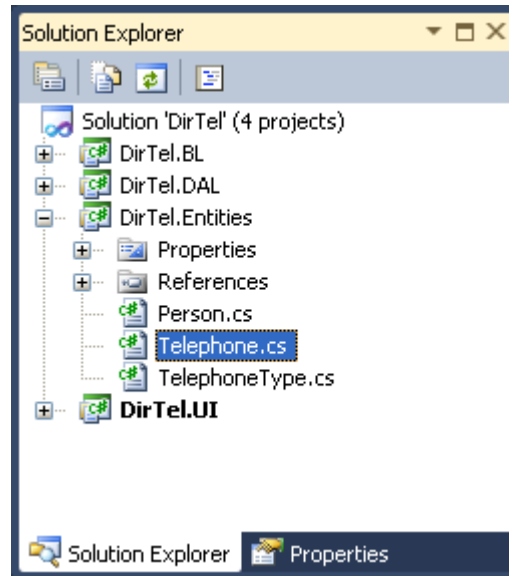
7. Cree el constructor por defecto y constructor parametrizado.

```
/// <summary>
/// Constructor por defecto
/// </summary>
public TelephoneType()
{
    // Asignar valores por defecto a los atributos
    this.value = TelephoneTypeCode.Home;
    this.name = TelephoneTypeCode.Home.ToString();
}

/// <summary>
/// Constructor parametrizado
/// </summary>
/// <param name="value">Valor correspondiente al tipo de telefono</param>
/// <param name="name">Descripción o nombre correspondiente</param>
public TelephoneType(TelephoneTypeCode value, string name)
{
    // Asignar los valores a los atributos
    this.value = value;
    this.name = name;
}
```

Telephone

8. Agregue una nueva clase Entidad para manejar los teléfonos, denominada **Telephone**.

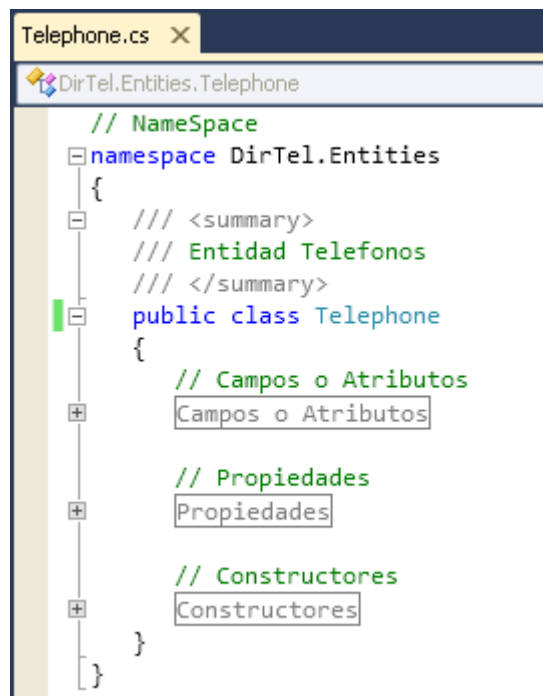


9. Edite el Contenido del Archivo **Telephone.cs** y adicione el encabezado de la documentación correspondiente.

```
Telephone.cs
DirTel.Entities.TelephoneType
#region Documentación
/*
 * Propiedad intelectual de Engineers & Tools (c).
 *
 * Unidad      : Telephone.cs
 * Descripcion : Entidad para manejar los datos de los telefonos
 * Autor       : Julio Cesar Robles Uribe - Jucer
 * Fecha       : 21-May-2010
 *
 * Fecha      Autor      Modificación
 * =====
 * 21-May-2010 Jucer      1 - Version Inicial
 */
#endregion Documentación

// Librerias
using System;
using System.Collections.Generic;
using System.Text;
```


10. Modifique el alcance de la clase a **public**, adicione la documentación y los comentarios correspondientes.



11. Agregue los campos y atributos de la siguiente forma:

```
// Identificador del Telefono
private long telephone_Id;
// Identificador de la persona
private long person_Id;
// Identificador del tipo de telefono
private TelephoneTypeCode telephone_Type;
// Numero del telefono
private string telephone_Number;
// Notas adicionales al telefono
private string notes;
```

12. Cree las propiedades así:

```
/// <summary>
/// Identificador del Telefono
/// </summary>
public long Telephone_Id
{
    get
    {
        return telephone_Id;
    }
    set
    {
        telephone_Id = value;
    }
}
```

```

/// <summary>
/// Identificador de la Persona
/// </summary>
public long Person_Id
{
    get
    {
        return person_Id;
    }
    set
    {
        person_Id = value;
    }
}

/// <summary>
/// Identificador del tipo de telefono
/// Hogar=1, Celular=2, Oficina=3
/// </summary>
public TelephoneTypeCode Telephone_Type
{
    get
    {
        return telephone_Type;
    }
    set
    {
        telephone_Type = value;
    }
}

/// <summary>
/// Numero del Telefono
/// </summary>
public string Telephone_Number
{
    get
    {
        return telephone_Number;
    }
    set
    {
        telephone_Number = value;
    }
}

/// <summary>
/// Notas adicionales al telefono
/// </summary>
public string Notes
{
    get
    {
        return notes;
    }
    set
    {
        notes = value;
    }
}

```

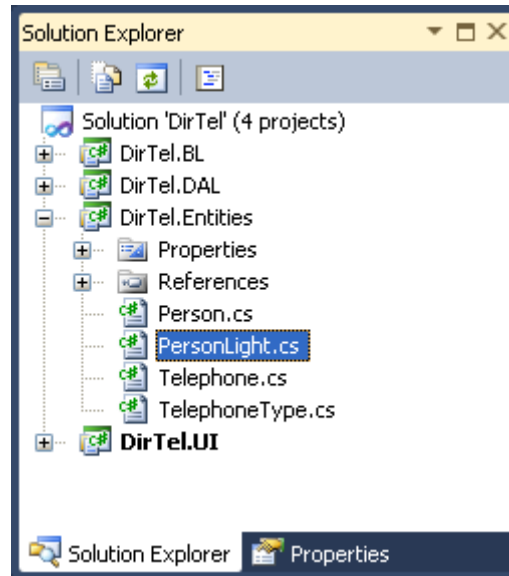
13. Cree el constructor por defecto y constructor parametrizado.

```
/// <summary>
/// Constructor por defecto
/// </summary>
public Telephone()
{
    // Identificador del Telefono
    telephone_Id = 0;
    // Identificador de la persona
    person_Id = 0;
    // Identificador del tipo de telefono
    telephone_Type = TelephoneTypeCode.Home;
    // Numero del telefono
    telephone_Number = string.Empty;
    // Notas adicionales al telefono
    notes = string.Empty;
}

/// <summary>
/// Constructor con parametros
/// </summary>
/// <param name="person_Id">Identificador de la Persona</param>
/// <param name="telephone_Type">Tipo de telefono (1-Hogar, 2-Movil, 3-Oficina </param>
/// <param name="telephone_Number">Numero del Telefono</param>
/// <param name="notes">Notas adicionales al telefono</param>
public Telephone(long person_Id, TelephoneTypeCode telephone_Type, string telephone_Number, string
notes)
{
    // Identificador de la persona
    this.person_Id = person_Id;
    // Identificador del tipo de telefono
    this.telephone_Type = telephone_Type;
    // Numero del telefono
    this.telephone_Number = telephone_Number;
    // Notas adicionales al telefono
    this.notes = notes;
}
```

PersonLight

14. Agregue una nueva clase Entidad para manejar los datos mínimos de una persona, denominada **PersonLight**.

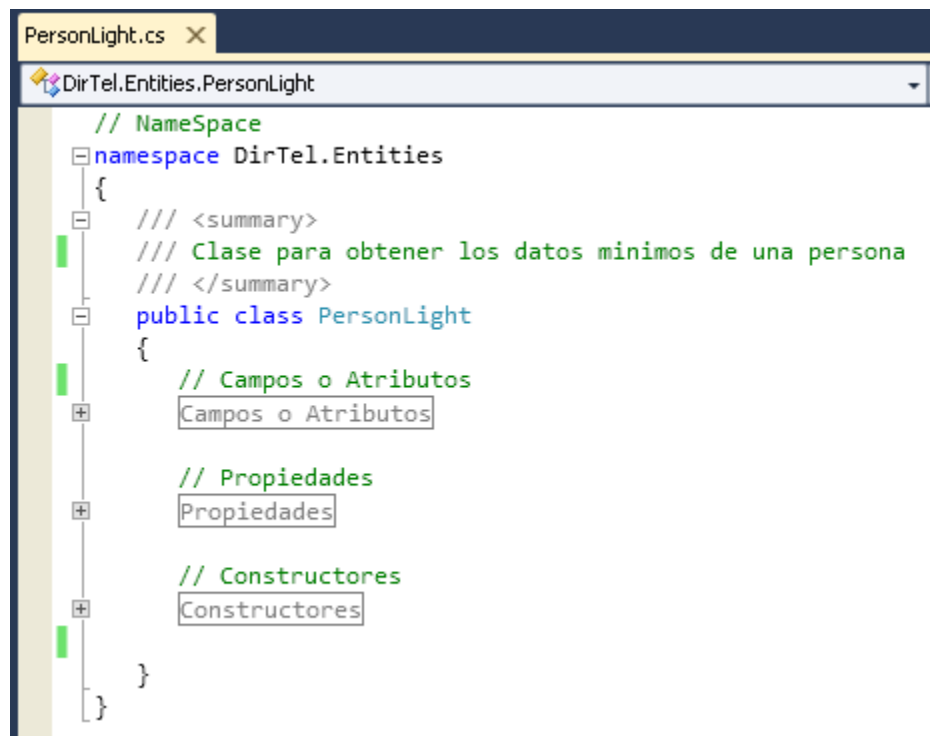


15. Edite el Contenido del Archivo **PersonLight.cs** y adicione el encabezado de la documentación correspondiente.

```
PersonLight.cs X
DirTel.Entities.PersonLight

#region Documentación
/*
 * Propiedad intelectual de Engineers & Tools (c).
 *
 * *****
 * Unidad      : PersonLight.cs
 * Descripción : Entidad para el manejo de los datos minimos de una persona (Id y Nombre)
 * Autor       : Julio Cesar Robles Uribe - Jucer
 * Fecha       : 21-May-2010
 *
 * Fecha      Autor      Modificación
 * =====
 * 21-May-2010 Jucer      1 - Version Inicial
 *
 * *****
 */
#endregion Documentación
```

16. Modifique el alcance de la clase a **public**, adicione la documentación y los comentarios correspondientes.



```
PersonLight.cs X
DirTel.Entities.PersonLight

// Namespace
namespace DirTel.Entities
{
    /// <summary>
    /// Clase para obtener los datos minimos de una persona
    /// </summary>
    public class PersonLight
    {
        // Campos o Atributos
        Campos o Atributos

        // Propiedades
        Propiedades

        // Constructores
        Constructores
    }
}
```

17. Agregue los campos y atributos de la siguiente forma:

```
// Identificador de la persona
private long person_Id;
// Nombre completo de la persona (Nombres y Apellidos)
private string fullName;
```

18. Cree las propiedades así:

```
/// <summary>
/// Identificador de la Persona
/// </summary>
public long Person_Id
{
    get
    {
        return person_Id;
    }
    set
    {
        person_Id = value;
    }
}
```

```

/// <summary>
/// Nombre completo de la persona (Nombres y Apellidos)
/// </summary>
public string FullName
{
    get
    {
        return fullName;
    }
    set
    {
        fullName = value;
    }
}

```

19. Cree el constructor por defecto y constructor parametrizado.

```

/// <summary>
/// Constructor por defecto
/// </summary>
public PersonLight()
{
    this.person_Id = 0;
    this.fullName = string.Empty;
}

/// <summary>
/// Constructor con parametros
/// </summary>
/// <param name="person_Id">Identificador de la persona</param>
/// <param name="fullName">Nombre completo de la persona (Nombres y Apellidos)</param>
public PersonLight(long person_Id, string fullName)
{
    this.person_Id = person_Id;
    this.fullName = fullName;
}

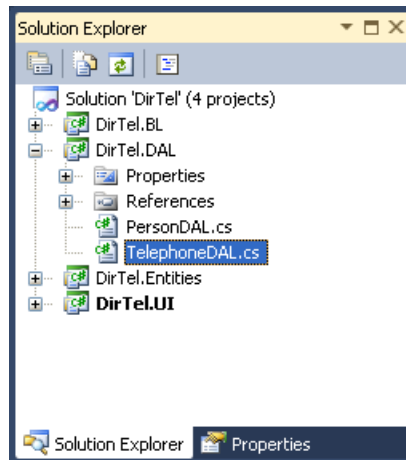
```

20. Guarde todos los cambios y compile el proyecto de entidades

Lógica de Acceso a Datos (DAL)

TelephoneDAL

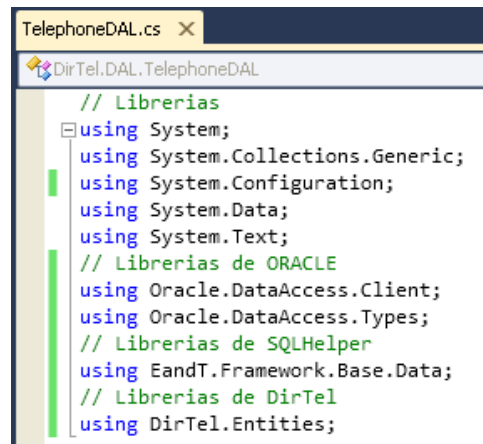
1. Adiciona una clase denominada **TelephoneDAL** al proyecto de Acceso a datos.



2. Edite el contenido del archivo **TelephoneDAL.cs** y adicione el encabezado de la documentación correspondiente.

```
TelephoneDAL.cs
DirTel.DAL.TelephoneDAL
#region Documentación
* Propiedad intelectual de Engineers & Tools (c).
* Unidad : TelephoneDAL.cs
* Descripción : Permite acceder a los datos de los telefonos
* Autor : Julio Cesar Robles Uribe - Jucer
* Fecha : 23-May-2010
*
* Fecha Autor Modificación
* =====
* 23-May-2010 Jucer 1 - Version Inicial
* =====
#endregion Documentación
```

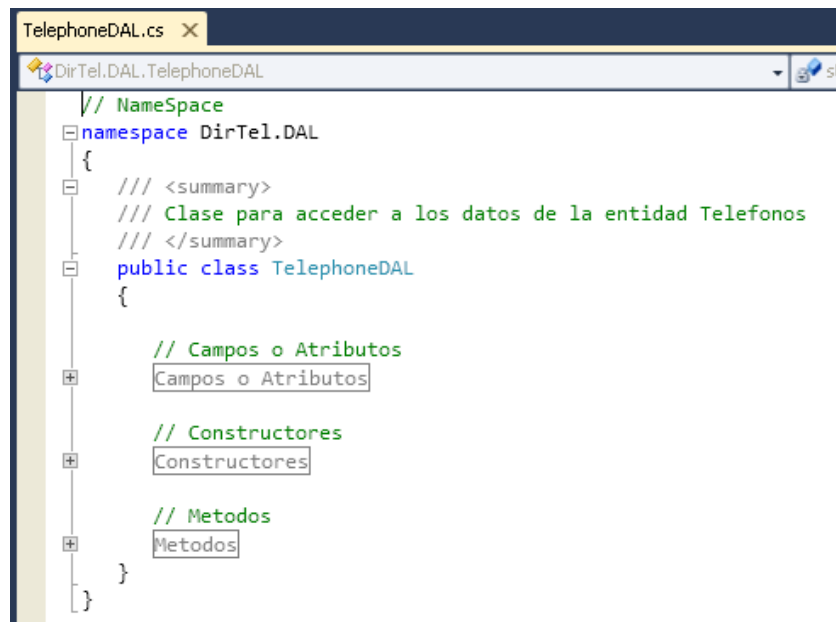
3. Adicione las librerías de ORACLE y de acceso a datos, así como las del manejo de entidades y de configuración.



```
TelephoneDAL.cs X
DirTel.DAL.TelephoneDAL

// Librerias
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Text;
// Librerias de ORACLE
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;
// Librerias de SQLHelper
using EandT.Framework.Base.Data;
// Librerias de DirTel
using DirTel.Entities;
```

4. Modifique el alcance de la clase a **public** y adicionar la documentación y adicione los valores correspondientes a la documentación.



```
TelephoneDAL.cs X
DirTel.DAL.TelephoneDAL

// Namespace
namespace DirTel.DAL
{
    /// <summary>
    /// Clase para acceder a los datos de la entidad Telefonos
    /// </summary>
    public class TelephoneDAL
    {
        // Campos o Atributos
        Campos o Atributos

        // Constructores
        Constructores

        // Metodos
        Metodos
    }
}
```

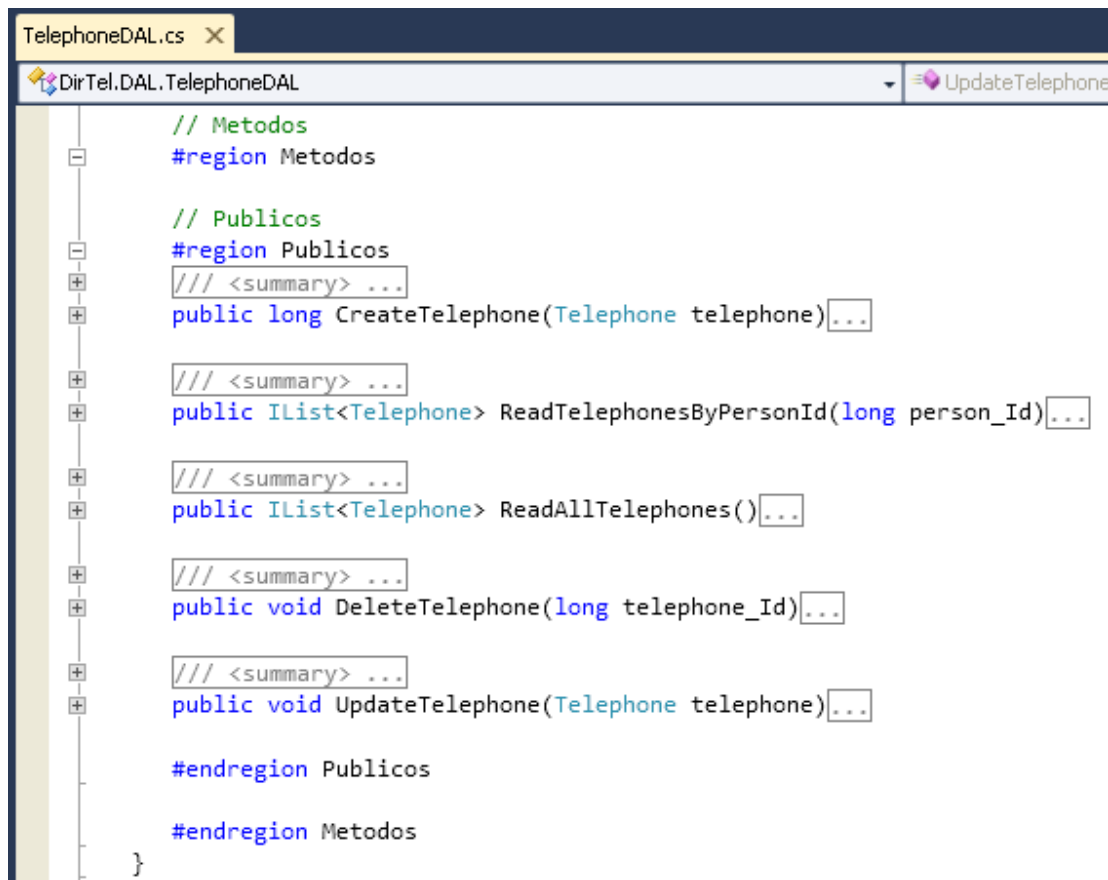
5. Adicione los campos a tributos para acceder a los datos así:

```
// Cadena de conexion a la Base de datos
private string strCnx;
// Variable para manejar el enlace a la base de datos
private DataBase db;
// Variable para manejar la conexion a la base de datos
private IDbConnection cnx;
```


6. Adicione el constructor por defecto que instancia la cadena de conexión a la base de datos así:

```
/// <summary>
/// Constructor por defecto
/// </summary>
public TelephoneDAL()
{
    // Inicializar la cadena de conexion
    strCnx =
    ConfigurationManager.ConnectionStrings["DirTelCnx"].ConnectionString;
}
```

7. Agregue los métodos para el manejo de los datos del teléfono así:



```
TelephoneDAL.cs X
DirTel.DAL.TelephoneDAL UpdateTelephone

// Metodos
#region Metodos

// Publicos
#region Publicos
/// <summary> ...
public long CreateTelephone(Telephone telephone)...

/// <summary> ...
public IList<Telephone> ReadTelephonesByPersonId(long person_Id)...

/// <summary> ...
public IList<Telephone> ReadAllTelephones()...

/// <summary> ...
public void DeleteTelephone(long telephone_Id)...

/// <summary> ...
public void UpdateTelephone(Telephone telephone)...

#endregion Publicos

#endregion Metodos
}
```

```

/// <summary>
/// Inserta los datos del telefono
/// </summary>
/// <param name="telephone">Datos del Telefono</param>
/// <returns>Identificador del Telefono</returns>
public long CreateTelephone(Telephone telephone)
{
    // Variable para retornar el id del telefono
    long telephone_Id = 0;

    // Establecer la conexion a la base de datos
    db = new DataBase(ProviderType.Oracle, strCnx);

    // Crear la Conexion
    cnx = db.CreateAndOpenConnection();

    // Crear el Comando
    IDbCommand cmd = db.CreateCommand(cnx);
    cmd.CommandText = "DA_TELEPHONES_PKG.CreateTelephone";
    cmd.CommandType = CommandType.StoredProcedure;
    // Asignar los parametros
    db.AddInParameter(cmd, "pm_Person_Id", telephone.Person_Id, DbType.Int64);
    db.AddInParameter(cmd, "pm_Telephone_Type",
Convert.ToInt32(telephone.Telephone_Type), DbType.Int32);
    db.AddInParameter(cmd, "pm_Telephone_Number", telephone.Telephone_Number,
DbType.String);
    db.AddInParameter(cmd, "pm_Notes", telephone.Notes, DbType.String);
    // Adicionar el parametro de salida
    db.AddOutParameter(cmd, "pm_Telephone_Id", null, DbType.Int64);

    // Ejecutar el procedimiento
    int rowsAffected = db.ExecuteNonQuery(cnx, cmd);

    // Referencia el parametro de salida para el Id del telefono
    OracleParameter outTelephone_Id = cmd.Parameters[db.ParameterToken() +
"pm_Telephone_Id"] as OracleParameter;

    // Obtener el valor del parametro
    telephone_Id = Convert.ToInt64(outTelephone_Id.Value);

    //Cerrar la conexion
    db.CloseConnection(cnx);

    // Retornar el valor
    return telephone_Id;
}

```

```

/// <summary>
/// Obtener todos los telefonos de una persona
/// </summary>
/// <param name="person_Id">Identificador de la persona</param>
/// <returns> La Lista de Telefonos</returns>
public IList<Telephone> ReadTelephonesByPersonId(long person_Id)
{
    // Crear la lista para retornar los datos
    IList<Telephone> lstTelephones = new List<Telephone>();

    // Establecer la conexion a la base de datos
    db = new DataBase(ProviderType.Oracle, strCnx);

    // Crear la Conexion
    cnx = db.CreateAndOpenConnection();

    // Crear el Comando especial para ORACLE
    IDbCommand cmd = db.CreateCommand(cnx);
    cmd.CommandText = "DA_TELEPHONES_PKG.ReadTelephonesByPersonId";
    cmd.CommandType = CommandType.StoredProcedure;

    // Asignar los parametros
    db.AddInParameter(cmd, "pm_Person_Id", person_Id, DbType.Int64);

    // Crear el Parametro enlazado al Cursor Referenciado
    OracleParameter paramRefCursor = db.AddCommandRefCurParameter(((OracleCommand)cmd),
"cur_Telephones", ParameterDirection.Output, null);

    // Ejecutar el query
    db.ExecuteNonQuery(cnx, cmd);

    // Obtener el Cursor Referenciado
    OracleRefCursor refCur = (OracleRefCursor)paramRefCursor.Value;
    // Obtener el DataReader desde el cursor referenciado
    OracleDataReader dr = refCur.GetDataReader();

    // Ciclo para Recorer los datos
    while (dr.Read())
    {
        // Variable Person para obtener los datos
        Telephone telephone = new Telephone();

        // Leer los valores
        telephone.Telephone_Id = Convert.ToInt64(dr["Telephone_Id"].ToString());
        telephone.Person_Id = Convert.ToInt64(dr["Person_Id"].ToString());
        telephone.Telephone_Type =
(TelephoneTypeCode)Convert.ToInt32(dr["Telephone_Type"].ToString());
        telephone.Telephone_Number = dr["Telephone_Number"].ToString();
        telephone.Notes = dr["Notes"].ToString();

        // Adicionar el Valor a la lista
        lstTelephones.Add(telephone);
    }

    // Cerar la Conexion
    db.CloseConnection(cnx);

    // retornar la lista
    return lstTelephones;
}

```

```

/// <summary>
/// Obtener todos los datos de los Telefonos
/// </summary>
/// <returns>Lista con los datos de los telefonos</returns>
public IList<Telephone> ReadAllTelephones()
{
    // Crear la lista para retornar los datos
    IList<Telephone> lstTelephones = new List<Telephone>();

    // Establecer la conexion a la base de datos
    db = new DataBase(ProviderType.Oracle, strCnx);

    // Crear la Conexion
    cnx = db.CreateAndOpenConnection();

    // Crear el Comando especial para ORACLE
    IDbCommand cmd = db.CreateCommand(cnx);
    cmd.CommandText = "DA_TELEPHONES_PKG.ReadAllTelephones";
    cmd.CommandType = CommandType.StoredProcedure;

    // Crear el Parametro enlazado al Cursor Referenciado
    OracleParameter paramRefCursor = db.AddCommandRefCurParameter(((OracleCommand)cmd),
"cur_Telephones", ParameterDirection.Output, null);

    // Ejecutar el query
    db.ExecuteNonQuery(cnx, cmd);

    // Obtener el Cursor Referenciado
    OracleRefCursor refCur = (OracleRefCursor)paramRefCursor.Value;
    // Obtener el DataReader desde el cursor referenciado
    OracleDataReader dr = refCur.GetDataReader();

    // Ciclo para Recorer los datos
    while (dr.Read())
    {
        // Variable Person para obtener los datos
        Telephone telephone = new Telephone();

        // Leer los valores
        telephone.Telephone_Id = Convert.ToInt64(dr["Telephone_Id"].ToString());
        telephone.Person_Id = Convert.ToInt64(dr["Person_Id"].ToString());
        telephone.Telephone_Type =
(TelephoneTypeCode)Convert.ToInt32(dr["Telephone_Type"].ToString());
        telephone.Telephone_Number = dr["Telephone_Number"].ToString();
        telephone.Notes = dr["Notes"].ToString();

        // Adicionar el Valor a la lista
        lstTelephones.Add(telephone);
    }

    // Cerar la Conexion
    db.CloseConnection(cnx);

    // retornar la lista
    return lstTelephones;
}

```

```

/// <summary>
/// Borrar el Registro de la tabla segun el Id
/// </summary>
/// <param name="telephone Id">Id a borrar</param>
public void DeleteTelephone(long telephone_Id)
{
    // Establecer la conexion a la base de datos
    db = new DataBase(ProviderType.Oracle, strCnx);

    // Crear la Conexion
    cnx = db.CreateAndOpenConnection();

    // Crear el Comando
    IDbCommand cmd = db.CreateCommand(cnx);
    cmd.CommandText = "DA_TELEPHONES_PKG.DeleteTelephone";
    cmd.CommandType = CommandType.StoredProcedure;
    // Asignar los parametros
    db.AddInParameter(cmd, "pm_Telephone_Id", telephone_Id, DbType.Int64);

    // Ejecutar el procedimiento
    int rowsAffected = db.ExecuteNonQuery(cnx, cmd);

    // Cerrar la conexion
    db.CloseConnection(cnx);
}

/// <summary>
/// Actualiza los datos del telefono
/// </summary>
/// <param name="telephone">Datos del telefono</param>
public void UpdateTelephone(Telephone telephone)
{
    // Establecer la conexion a la base de datos
    db = new DataBase(ProviderType.Oracle, strCnx);

    // Crear la Conexion
    cnx = db.CreateAndOpenConnection();

    // Crear el Comando
    IDbCommand cmd = db.CreateCommand(cnx);
    cmd.CommandText = "DA_TELEPHONES_PKG.UpdateTelephone";
    cmd.CommandType = CommandType.StoredProcedure;
    // Asignar los parametros
    db.AddInParameter(cmd, "pm_Telephone_Id", telephone.Telephone_Id, DbType.Int64);
    db.AddInParameter(cmd, "pm_Person_Id", telephone.Person_Id, DbType.Int64);
    db.AddInParameter(cmd, "pm_Telephone_Type",
Convert.ToInt32(telephone.Telephone_Type), DbType.Int32);
    db.AddInParameter(cmd, "pm_Telephone_Number", telephone.Telephone_Number,
DbType.String);
    db.AddInParameter(cmd, "pm_Notes", telephone.Notes, DbType.String);

    // Ejecutar el procedimiento
    int rowsAffected = db.ExecuteNonQuery(cnx, cmd);

    // Cerrar la conexion
    db.CloseConnection(cnx);
}

```

PersonDAL

8. Modifique la clase **PersonDAL**, para incluir el método **ReadAllPersonsLight** así:

```
/// <summary>
/// Obtener los datos minimos de la persona (Id y Nombre Completo)
/// </summary>
/// <returns>Lista con los datos minimos</returns>
public IList<PersonLight> ReadAllPersonsLight()
{
    // Crear la lista para retornar los datos
    IList<PersonLight> lstPersonsLight = new List<PersonLight>();

    // Establecer la conexion a la base de datos
    db = new DataBase(ProviderType.Oracle, strCnx);

    // Crear la Conexion
    cnx = db.CreateAndOpenConnection();

    // Crear el Comando
    IDbCommand cmd = db.CreateCommand(cnx);
    cmd.CommandText = "DA_PERSONS_PKG.ReadAllPersonsLight";
    cmd.CommandType = CommandType.StoredProcedure;

    // Crear el Parametro enlazado al Cursor Referenciado
    OracleParameter paramRefCursor = db.AddCommandRefCurParameter((OracleCommand)cmd,
"cur_Persons", ParameterDirection.Output, null);

    // Ejecutar el query
    db.ExecuteNonQuery(cnx, cmd);

    // Obtener el Cursor Referenciado
    OracleRefCursor refCur = (OracleRefCursor)paramRefCursor.Value;
    // Obtener el DataReader desde el cursor referenciado
    OracleDataReader dr = refCur.GetDataReader();

    // Ciclo para Recorer los datos
    while (dr.Read())
    {
        // Variable Person para obtener los datos
        PersonLight perLigth = new PersonLight();

        // Leer los valores
        perLigth.Person_Id = Convert.ToInt64(dr["Person_Id"].ToString());
        perLigth.FullName = dr["FullName"].ToString();

        // Adicionar el Valor a la lista
        lstPersonsLight.Add(perLigth);
    }

    // Cerar la Conexion
    db.CloseConnection(cnx);

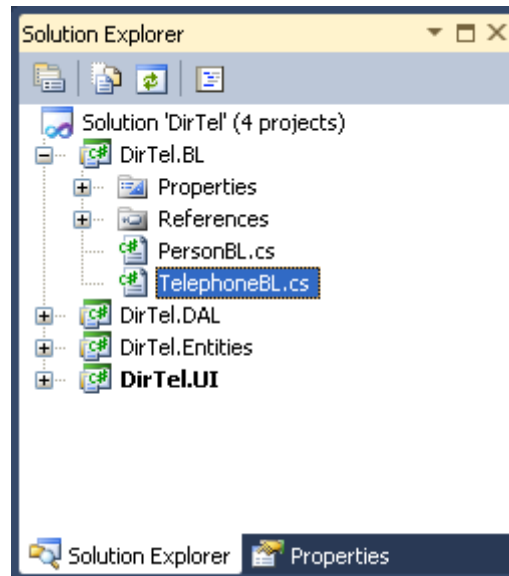
    // retornar la lista
    return lstPersonsLight;
}
```

9. Guarde todos los cambios y compile el proyecto de lógica de acceso a datos.

Lógica de Negocios (BL)

TelephoneBL

1. Adicione la clase **TelephoneBL** al proyecto de lógica de negocios.



2. Edite el contenido del archivo **TelephoneBL.cs** y adicione el encabezado de la documentación correspondiente.

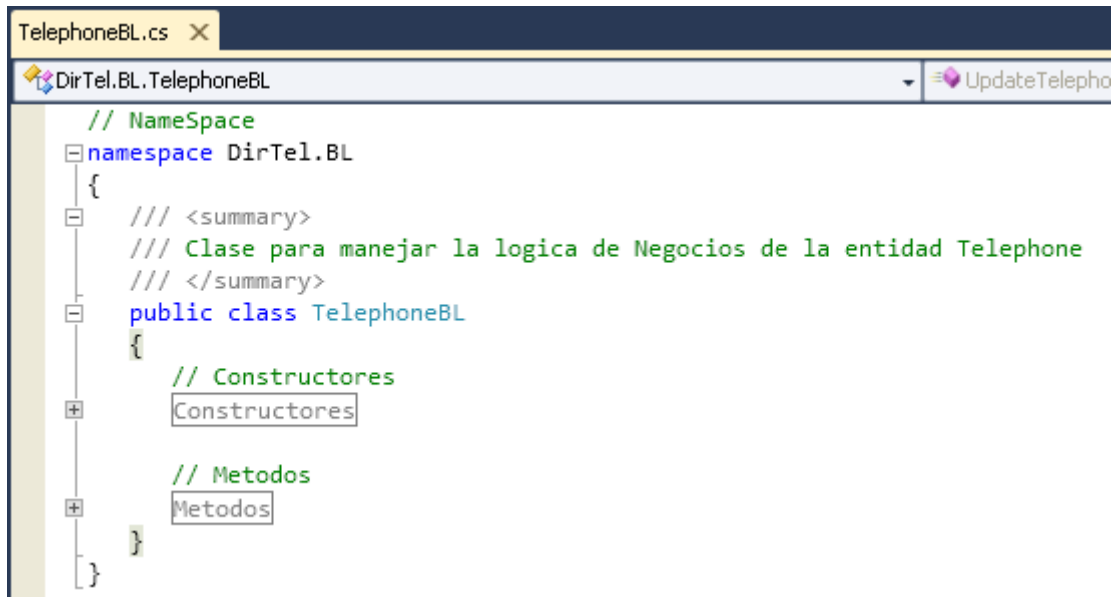
```
TelephoneBL.cs
DirTel.BL.TelephoneBL
TelephoneBL()

#region Documentación
*****
* Propiedad intelectual de Engineers & Tools (c).
*****
* Unidad      : TelephoneBL.cs
* Descripción  : Clase para Manejar la Logica de Negocios para la entidad Telephone
* Autor       : Julio Cesar Robles Uribe - Jucer
* Fecha      : 24-May-2010
*
* Fecha      Autor      Modificación
* =====
* 24-May-2010 Jucer      1 - Version Inicial
*****
#endregion Documentación
```

3. Adicione las librerías de de acceso a datos, así como las del manejo de entidades.

```
// Librerías de DirTel
using DirTel.Entities;
using DirTel.DAL;
```

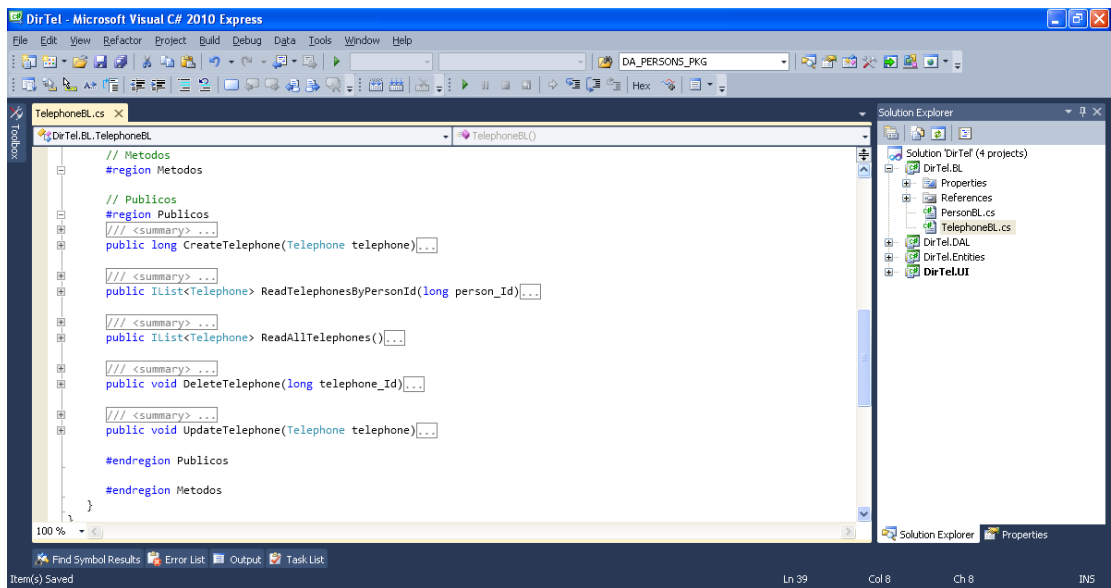
4. Modifique el alcance de la clase a **public** y adicionar la documentación y adicione los valores correspondientes a la documentación.



5. Adicione el constructor por defecto.

```
/// <summary>
/// Constructor por defecto
/// </summary>
public TelephoneBL()
{
}
```

6. Adicione los métodos para el manejo de la lógica así:




```

/// <summary>
/// Inserta los datos del telefono
/// </summary>
/// <param name="telephone">Datos del Telefono</param>
/// <returns>Identificador del Telefono</returns>
public long CreateTelephone(Telephone telephone)
{
    // Crear el Objeto de Datos
    TelephoneDAL telephoneDAL = new TelephoneDAL();

    // Ejecutar el Comando y retornar el id insertado
    long telephone_Id = telephoneDAL.CreateTelephone(telephone);

    // retornar el valor
    return telephone_Id;
}

/// <summary>
/// Obtener todos los telefonos de una persona
/// </summary>
/// <param name="person_Id">Identificador de la persona</param>
/// <returns>Lista de Telefonos</returns>
public IList<Telephone> ReadTelephonesByPersonId(long person_Id)
{
    // Crear el Objeto de Datos
    TelephoneDAL telephoneDAL = new TelephoneDAL();

    // Ejecutar el Comando
    IList<Telephone> lstTelephones = telephoneDAL.ReadTelephonesByPersonId(person_Id);

    // retornar la lista
    return lstTelephones;
}

/// <summary>
/// Obtener todos los datos de los Telefonos
/// </summary>
/// <returns>Lista con los datos de los telefonos</returns>
public IList<Telephone> ReadAllTelephones()
{
    // Crear el Objeto de Datos
    TelephoneDAL telephoneDAL = new TelephoneDAL();

    // Ejecutar el Comando
    IList<Telephone> lstTelephones = telephoneDAL.ReadAllTelephones();

    // retornar la lista
    return lstTelephones;
}

/// <summary>
/// Borrar el Registro de la tabla segun el Id
/// </summary>
/// <param name="telephone_Id">Id a borrar</param>
public void DeleteTelephone(long telephone_Id)
{
    // Crear el Objeto de Datos
    TelephoneDAL telephoneDAL = new TelephoneDAL();

    // Ejecutar el Comando
    telephoneDAL.DeleteTelephone(telephone_Id);
}

```

```

/// <summary>
/// Actualiza los datos del telefono
/// </summary>
/// <param name="telephone">Datos del telefono</param>
public void UpdateTelephone(Telephone telephone)
{
    // Crear el Objeto de Datos
    TelephoneDAL personDAL = new TelephoneDAL();

    // Ejecutar el Comando
    personDAL.UpdateTelephone(telephone);
}

```

PersonBL

7. Modifique el archivo **PersonBL.cs** y adicione el método **ReadAllPersonsLight** así:

```

/// <summary>
/// Obtener los datos minimos de las personas
/// </summary>
/// <returns>Lista de Id y Nombres completos (Nombres y Apellidos)</returns>
public IList<PersonLight> ReadAllPersonsLight()
{
    // Crear el Objeto de Datos
    PersonDAL personDAL = new PersonDAL();

    // Ejecutar el Comando
    IList<PersonLight> lstPersonsLight = personDAL.ReadAllPersonsLight();

    return lstPersonsLight;
}

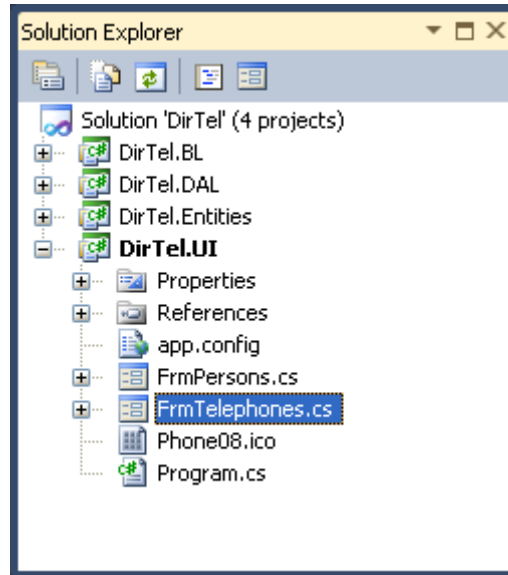
```

8. Guarde todos los cambios y compile el proyecto de lógica de negocios.

Interfaz de Usuario (UI)

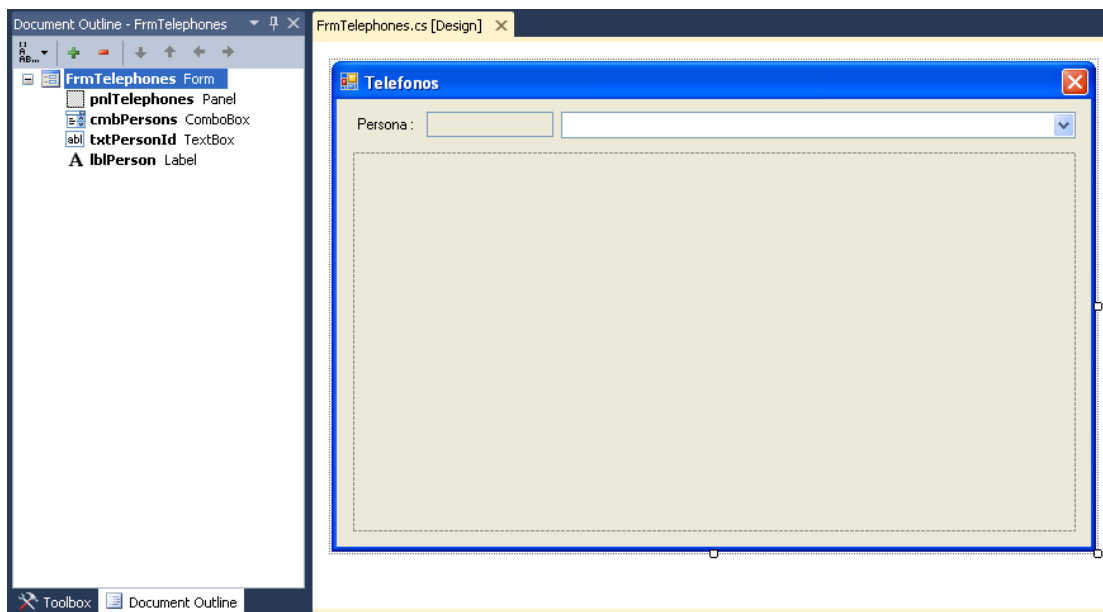
FrmTelephones

1. Adicione un nuevo formulario denominado FrmTelephones al proyecto de DirTel.UI



2. Modifique el formulario **FrmTelephones.cs** y cambie el valor de las siguientes propiedades:
 - a. **AutoSizeMode**=GrowAndShrink (No permitir que se redimensione)
 - b. **MaximizeBox**=False (Quitar el botón de Maximizar)
 - c. **MinimizeBox**=False (Quitar el botón de Minimizar)
 - d. **Text**=Telefonos (Titulo del Formulario)
 - e. **StartPosition**=CenterScreen (Poner centrado el formulario)
3. Adicione los siguientes controles y modifique sus propiedades así:
 - a. **Label**
 - i. **Name** = lblPerson
 - ii. **Text** = Persona :
 - b. **TextBox**
 - i. **Name** = txtPersonId
 - ii. **ReadOnly** = True
 - c. **ComboBox** sss
 - i. **Name** = cmbPersons
 - ii. **DropDownStyle** = DropDownList
 - d. **Panel**
 - i. **Name** = pnlTelephones

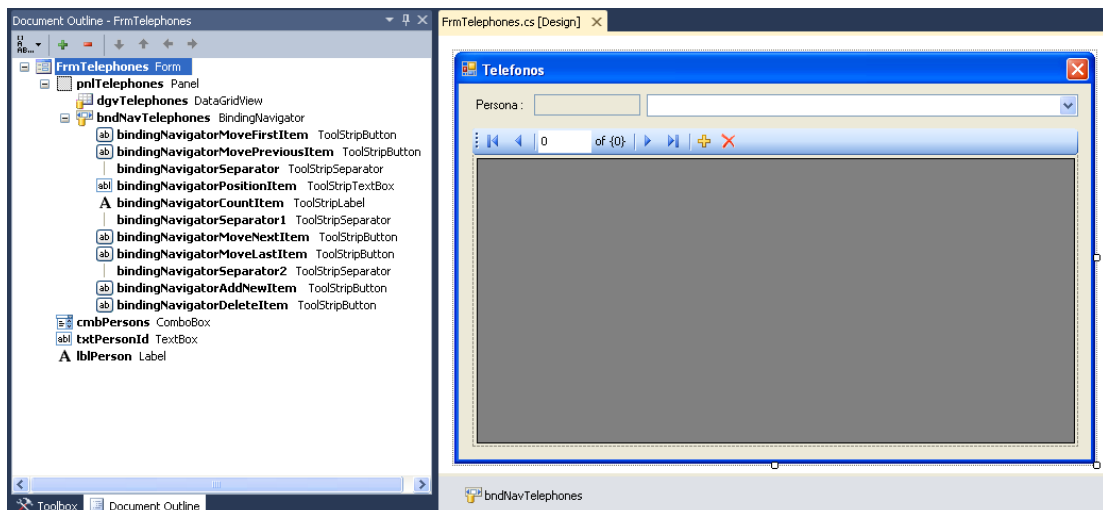
4. Reorganice los controles, su distribución y tamaño del formulario para dejarlo de la siguiente forma:



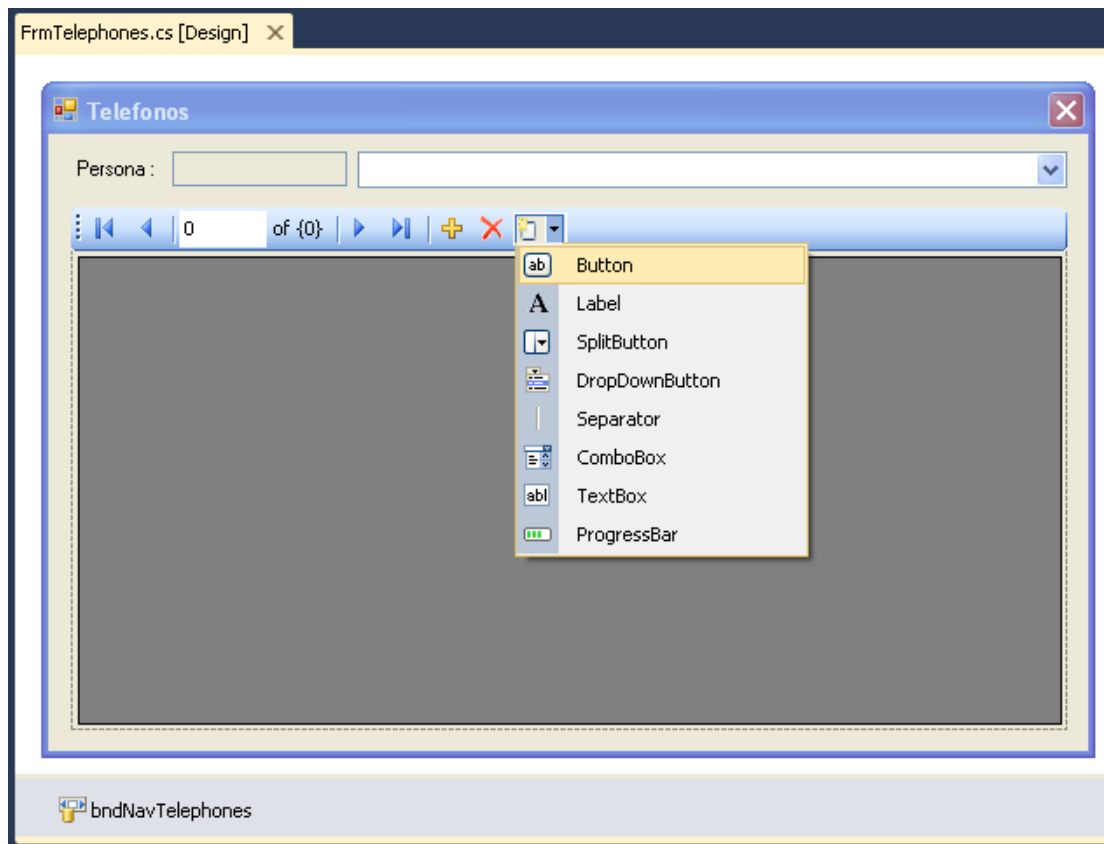
5. Seleccionando el control de panel adicione los siguientes controles y modifique sus propiedades así:

- a. **BindingNavigator**
 - i. **Name** = bndNavTelephones
- b. **DataGridView**
 - i. **Name** = dgvTelephones

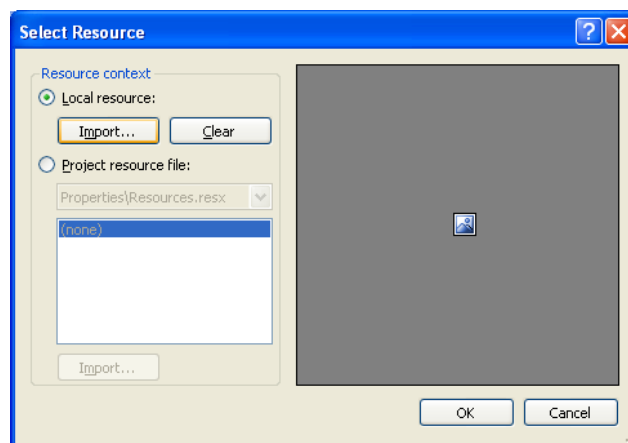
6. Modifique el tamaño de los controles para redistribuir su tamaño así:



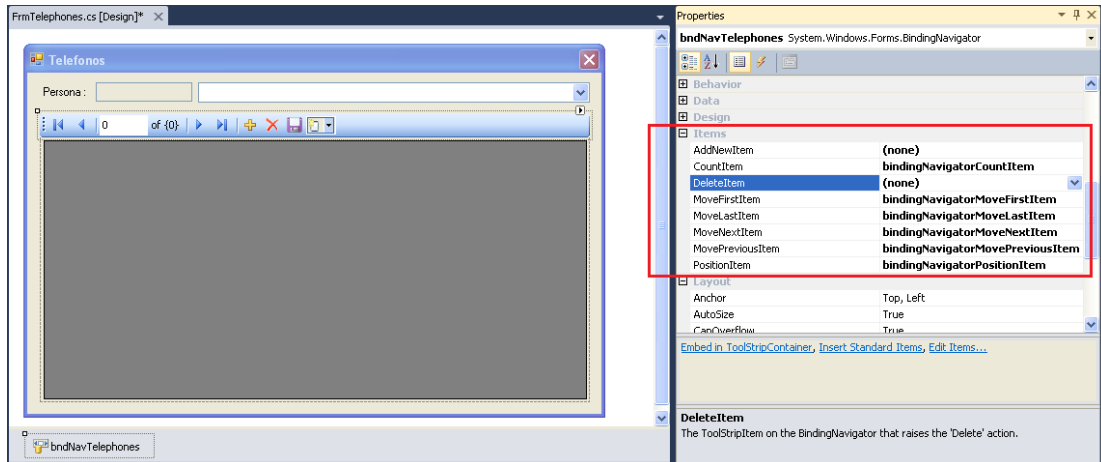
7. Seleccione el **BindingNavigator** para adicionar un nuevo botón así:



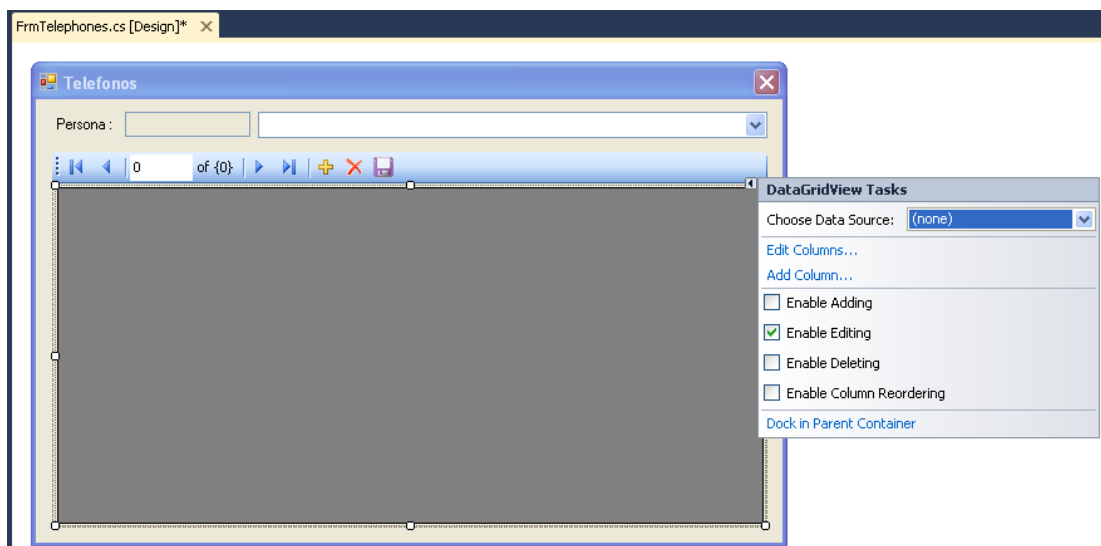
8. Cambiamos las propiedades de este nuevo botón de la siguiente forma:
- a. **Name**= bindingNavigatorSaveItems
 - b. **Text**= SaveItems
 - c. **ToolTipText**= Save Items
9. Seleccionamos el nuevo botón y desde la ventana de propiedades escogemos la propiedad **Image**, para lo cual se despliega una ventana como la siguiente:



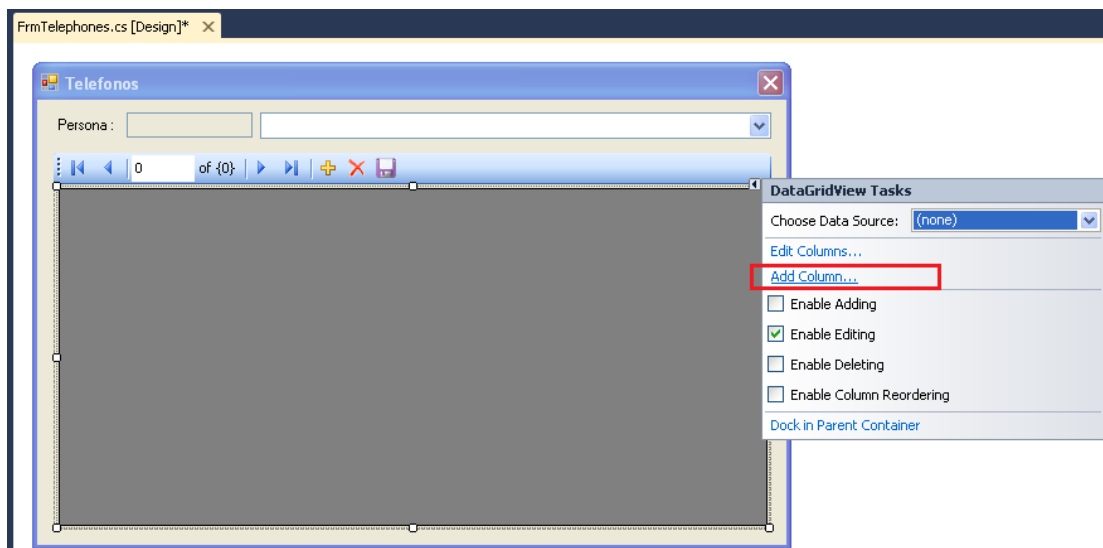
10. Escogemos la opción de importar del contexto local para que la imagen quede asociada al formulario y no al proyecto, esto nos facilita que al copiar el formulario a otra solución o proyecto este se vaya con la imagen incrustada dentro del formulario.
11. Escogemos del directorio de **C:\Proyecto\SupportData\Recursos\Png**, la imagen de **bindingNavigatorSaveItem.Image.png** y aceptamos el nuevo recurso
12. Ahora seleccionando nuevamente todo el Navegador (**bndNavTelephones**) vamos a modificarlo eliminando las funciones de adición y eliminación ya que se van a personalizar en este ejercicio. Para facilitar esta modificación organizamos las propiedades por categorías.



13. Cambiamos su valor a ninguno (none), únicamente a las propiedades de **AddNewItem** y a la **DeleteItem**.
14. Ahora modificamos la rejilla (Grid) para no permitir la adición ni el borrado de los registros por medio la rejilla adicionar las columnas correspondientes a los datos de la entidad **Telephone** así:



15. Ahora adicione las columnas a la Grid, seleccionando la opción Adicionar columna (Add Column ...)

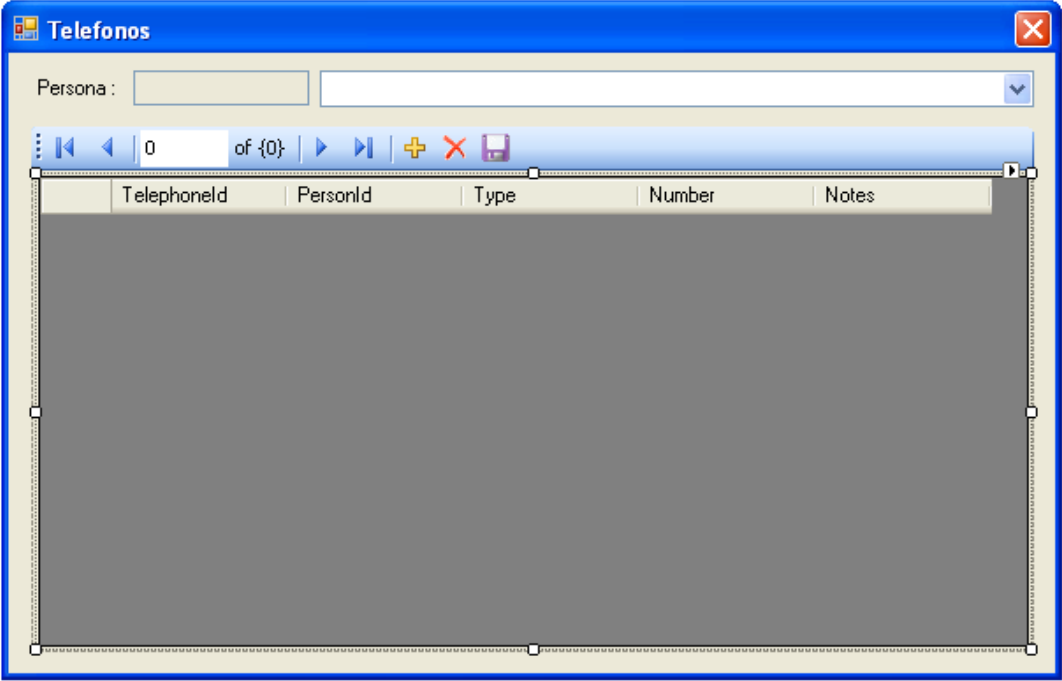


16. Asigne los nombres, tipos y títulos

The 'Add Column' dialog box is shown. It has two radio buttons: 'Databound column' (unselected) and 'Unbound column' (selected). Under 'Databound column', there is a section 'Columns in the DataSource' with an empty list box. Under 'Unbound column', there are three text boxes: 'Name' with 'ColumnTelephoneId', 'Type' with 'DataGridViewTextBoxColumn', and 'Header text' with 'TelephoneId'. Below these are three checkboxes: 'Visible' (checked), 'Read Only' (unchecked), and 'Frozen' (unchecked). At the bottom are 'Add' and 'Cancel' buttons.

Nombre (Name)	Tipo (Type)	Título (Header text)
ColumnTelephoneId	DataGridViewTextBoxColumn	TelephoneId
ColumnPersonId	DataGridViewTextBoxColumn	PersonId
ColumnTelephoneType	DataGridViewComboBoxColumn	Type
ColumnTelephoneNumber	DataGridViewTextBoxColumn	Number
ColumnTelephoneNotes	DataGridViewTextBoxColumn	Notes

17. Al terminar adicionar las columnas debe obtener lo siguiente:



FormTitlephones.cs [Design] X

Telefonos

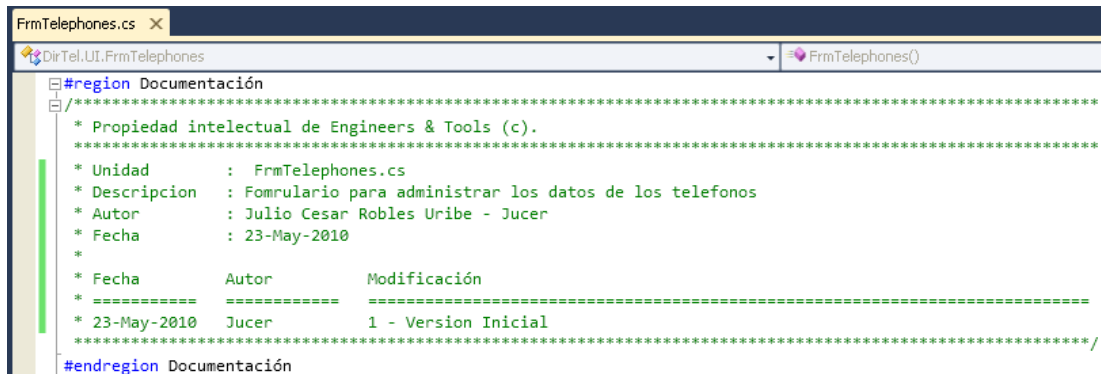
Persona :

0 of {0}

TelephoneId	PersonId	Type	Number	Notes
-------------	----------	------	--------	-------

18. Ahora modifiquemos el código del formulario para ello presione la tecla F7.

19. Modifique el encabezado de la documentación del formulario:

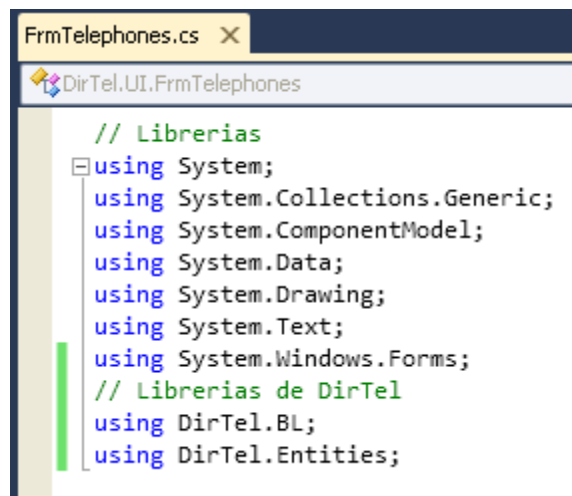


FrmTelephones.cs X

DirTel.UI.FrmTelephones FrmTelephones()

```
#region Documentación
* Propiedad intelectual de Engineers & Tools (c).
*
* Unidad      : FrmTelephones.cs
* Descripcion : Fomrulario para administrar los datos de los telefonos
* Autor       : Julio Cesar Robles Uribe - Jucer
* Fecha       : 23-May-2010
*
* Fecha      Autor      Modificación
* =====
* 23-May-2010 Jucer      1 - Version Inicial
#endregion Documentación
```

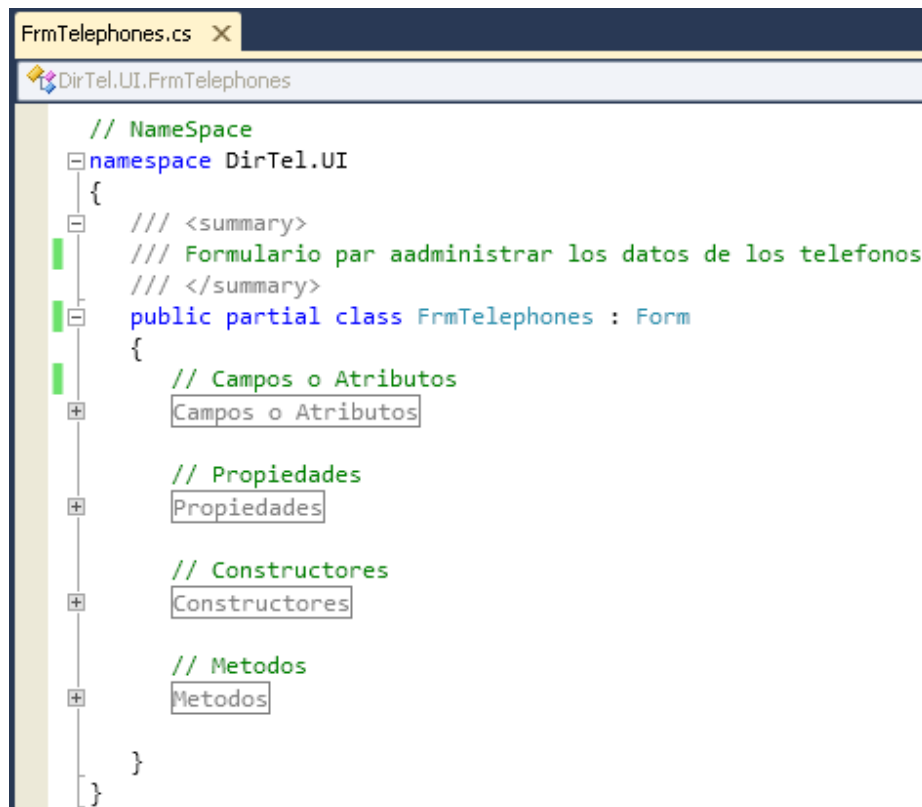

20. Adicione la referencia a las librerías de lógica de negocio (**DirTel.BL**) y de entidades (**DirTel.Entities**).



```
FrmTelephones.cs X
DirTel.UI.FrmTelephones

// Librerias
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
// Librerias de DirTel
using DirTel.BL;
using DirTel.Entities;
```

21. Cambie el alcance del formulario a **public** y documente su respectiva función.



```
FrmTelephones.cs X
DirTel.UI.FrmTelephones

// Namespace
namespace DirTel.UI
{
    /// <summary>
    /// Formulario par aadministrar los datos de los telefonos
    /// </summary>
    public partial class FrmTelephones : Form
    {
        // Campos o Atributos
        Campos o Atributos

        // Propiedades
        Propiedades

        // Constructores
        Constructores

        // Metodos
        Metodos
    }
}
```

22. Adicione el atributo o campo **person_Id**.

```
// Identificador de la persona que se referencia desde FrmPerson o desde el Menu
private long person_Id;
```

23. Adicione la propiedad `Person_Id`, que haga referencia al atributo `person_Id`.

```
/// <summary>
/// Identificador de la persona referenciado desde FrmPerson
/// si no se agina valor, su asignacion sera cero (0) por defecto
/// </summary>
public long Person_Id
{
    get
    {
        return person_Id;
    }
    set
    {
        person_Id = value;
    }
}
```

24. En el constructor asigne al atributo **`person_Id`** el valor de cero (0) por defecto.

```
/// <summary>
/// Construcor por defecto
/// </summary>
public FrmTelephones()
{
    InitializeComponent();

    // Asignar cero (0) por defecto al identificador de la persona
    this.person_Id = 0;
}
```

25. En la región de los métodos privados adicione el código siguiente:

```
// Metodos
#region Metodos

// Privados
#region Privados
/// <summary>
/// Cargar el formulario
/// </summary>
/// <param name="sender">Form</param>
/// <param name="e">Load</param>
private void FrmTelephones_Load(object sender, EventArgs e)
{
    // Cargar el combo con los datos de las personas
    this.LoadPersons(this.cmbPersons);

    // Enlazar el PersonId
    this.BindPersonId(this.txtPersonId, this.cmbPersons);

    // Desactivar la generacion automatica de las columnas para la Grid
    this.dgvTelephones.AutoGenerateColumns = false;

    // Inicializar las columnas de la Grid
    this.InitializeColumnnsTelephones();

    // Validar el identificador de la persona
    if (this.person_Id != 0)
    {
        // Seleccionar el valor del combo correspondiente
        cmbPersons.SelectedValue = this.person_Id;
    }

    // Obtener los datos de los telefonos
    this.LoadTelephones(this.cmbPersons, this.dgvTelephones);
}

/// <summary>
/// Ejecuta el cambio de registro al seleccionar un valor del combo
/// </summary>
/// <param name="sender">cmbPersons</param>
/// <param name="e">SelectionChangeCommitted</param>
private void cmbPersons_SelectionChangeCommitted(object sender, EventArgs e)
{
    // Variabla para referencia el combo
    ComboBox cmb = (ComboBox)sender;

    // Cargar los datos de los telefonos
    this.LoadTelephones(cmb, this.dgvTelephones);
}
```

```

/// <summary>
/// Controlar el borrado del registro
/// </summary>
/// <param name="sender">Binding Navigator</param>
/// <param name="e">DeleteItem</param>
private void bindingNavigatorDeleteItem_Click(object sender, EventArgs e)
{
    // Validar que existan registros para borrar
    if (dgvTelephones.RowCount > 0)
    {
        // Preguntar si en realidad lo desea eliminar
        DialogResult dlgResult = MessageBox.Show("Esta seguro de eliminar este
registro?", "Pregunta", MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button2);

        // Validar la respuesta
        if (dlgResult == DialogResult.Yes)
        {
            // Obtener el registro activo
            Telephone telephone =
(Telephone)dgvTelephones.CurrentRow.DataBoundItem;

            // Eliminar el registro
            this.DeleteTelephone(telephone.Telephone_Id);

            // Eliminar el registro de la grid
            BindingSource bndSrc = (BindingSource)dgvTelephones.DataSource;
            bndSrc.RemoveCurrent();
        }
    }
}

/// <summary>
/// Adiciona un registro a los datos de los telefonos
/// </summary>
/// <param name="sender">Binding Navigator</param>
/// <param name="e">AddNewItem</param>
private void bindingNavigatorAddNewItem_Click(object sender, EventArgs e)
{
    // Enlazar el Binding Source
    BindingSource bndSrc = (BindingSource)dgvTelephones.DataSource;

    // Instanciar una nuevo objeto de tipo Telephone
    Telephone telephone = new Telephone();

    // Obtener los datos de la persona seleccionada
    PersonLight personLight = (PersonLight)cmbPersons.SelectedItem;

    // asignar el Id de la persona
    telephone.Person_Id = personLight.Person_Id;
    // Asignar por defecto un telefono vacio
    telephone.Telephone_Number = String.Empty;

    // Adicionar un nuevo registro al Binding Source
    bndSrc.Add(telephone);
}

```

```

/// <summary>
/// Elimina el registro del telefono asociado
/// </summary>
/// <param name="telephone_Id">Identificador del telefono</param>
private void DeleteTelephone(long telephone_Id)
{
    // Instanciar objeto de logica de negocio
    TelephoneBL telephoneBL = new TelephoneBL();

    // Llamar al metodo de eliminar
    telephoneBL.DeleteTelephone(telephone_Id);
}

/// <summary>
/// Grabar los cambios
/// </summary>
/// <param name="sender">Binding Navigator</param>
/// <param name="e">SaveItems</param>
private void bindingNavigatorSaveItems_Click(object sender, EventArgs e)
{
    // Cambiar el foco al identificador de la persona
    txtPersonId.Focus();

    // Obtener la cantidad de filas
    int countRows = dgvTelephones.RowCount;

    // Validar la cantidad de filas
    if (countRows > 0)
    {
        // Objeto para referenciar el telefono activo
        Telephone telephoneCurrent = null;

        // Ciclo para recorrer la Grid
        for (int i = 0; i < countRows; i++)
        {
            // Seleccionar y obtener la fila
            DataGridViewRow dgvRow = dgvTelephones.Rows[i];

            // Obtener los datos del telefono actual
            telephoneCurrent = (Telephone)dgvRow.DataBoundItem;

            // Validar la fila actual
            if (telephoneCurrent.Telephone_Id.Equals(0))
            {
                // Insertar un nuevo registro
                this.CreateTelephone(telephoneCurrent);
            }
            else
            {
                // Actualizar el registro
                this.UpdateTelephone(telephoneCurrent);
            }
        }

        // Mostrar mensaje de datos actualizados
        MessageBox.Show("Datos Actualizados", "Mensaje", MessageBoxButtons.OK,
        MessageBoxIcon.Information);
    }
}

```

```

/// <summary>
/// Crear un nuevo registro
/// </summary>
/// <param name="telephoneCurrent">Datos del Registro a crear</param>
private void CreateTelephone(Telephone telephoneCurrent)
{
    // Instanciar objeto de logica de Negocio
    TelephoneBL telephoneBL = new TelephoneBL();

    // Insertar el registro y retornar el Id
    long telephoneId = telephoneBL.CreateTelephone(telephoneCurrent);

    // Asignar el identificaro del telefono
    telephoneCurrent.Telephone_Id = telephoneId;
}

/// <summary>
/// Actualiza el registro actual
/// </summary>
/// <param name="telephoneCurrent">Datos del registro a actualizar</param>
private void UpdateTelephone(Telephone telephoneCurrent)
{
    // Instanciar objeto de logica de Negocio
    TelephoneBL telephoneBL = new TelephoneBL();

    // Insertar el registro y retornar el Id
    telephoneBL.UpdateTelephone(telephoneCurrent);
}

/// <summary>
/// Cargar los datos de los telefonos según el valor de la persona
/// </summary>
/// <param name="comboBox">Combo de personas</param>
/// <param name="dataGridView">Grid de telefonos</param>
private void LoadTelephones(ComboBox comboBox, DataGridView dataGridView)
{
    // Obtener los datos de la persona seleccionada
    PersonLight personLight = (PersonLight)comboBox.SelectedItem;

    // Fuente de datos enlazada
    BindingSource bndSrc = new BindingSource();

    // Obtener el Id de la persona validando si hay datos
    long person_Id = (personLight == null)? 0 : personLight.Person_Id;

    // Obtener los Telefonos asociados a la persona
    bndSrc.DataSource = this.GetTelephones(person_Id);

    // Asignar la fuente de datos
    dataGridView.DataSource = bndSrc;

    // Asignar el origen de datos del Navegador a el correspondiente de la Grid
    bndNavTelephones.BindingSource = bndSrc;
}

```

```

/// <summary>
/// Obtener la lista de telefonos segun la persona
/// </summary>
/// <param name="person_Id">Identificador de la persona</param>
/// <returns>Lista de telefonos de la persona</returns>
private IList<Telephone> GetTelephones(long person_Id)
{
    // Instanciar el objeto de negocio
    TelephoneBL telephoneBL = new TelephoneBL();

    // Obtener la lista de telefonos por persona
    IList<Telephone> lstTelephones =
telephoneBL.ReadTelephonesByPersonId(person_Id);

    // Retornar la lista
    return lstTelephones;
}

/// <summary>
/// Inicializa las columnas de la grid
/// </summary>
private void InitializeColumnnsTelephones()
{
    // Asignar las propiedades a enlazar de las columnas con la entidad Telephone
    this.ColumnTelephoneId.DataPropertyName = "Telephone_Id";
    this.ColumnPersonId.DataPropertyName = "Person_Id";
    this.ColumnTelephoneType.DataPropertyName = "Telephone_Type";
    this.ColumnTelephoneNumber.DataPropertyName = "Telephone_Number";
    this.ColumnTelephoneNotes.DataPropertyName = "Notes";

    // Llenar el combo Box de Tipos de telefonos de Telephone_Type
    this.ColumnTelephoneType.DataSource = this.GetTelephoneTypes();

    // Enlazar las propiedades de la entidad TelephoneType con los de la columna
    this.ColumnTelephoneType.DisplayMember = "Name";
    this.ColumnTelephoneType.ValueMember = "Value";
}

```

```

/// <summary>
/// Obtiene la lista de tipos de telefonos
/// </summary>
/// <returns>Lista de tipos de telefonos (Codigo, Nombre)</returns>
private IList<TelephoneType> GetTelephoneTypes()
{
    // Variable para la lista de Tipos de telefonos
    IList<TelephoneType> lstTelephoneTypes = new List<TelephoneType>();

    // Obtener los nombres de la enumeracion
    string[] arrTelephoneTypeNames = Enum.GetNames(typeof(TelephoneTypeCode));

    // Obtener los codigos de la enumeracion
    int[] arrTelephoneTypeCodes =
(int[])Enum.GetValues(typeof(TelephoneTypeCode));

    // Obtener la longitud del arreglo de codigos
    int lenTelephoneTypes = arrTelephoneTypeCodes.Length;

    // Recorrer los arreglos, obtener los valores y adicionarlos a la lista
    for (int i = 0; i < lenTelephoneTypes; i++)
    {
        // Variable para el tipo de telefono
        TelephoneType telephoneType = new TelephoneType();

        // Asignar los valores
        telephoneType.Value = (TelephoneTypeCode)arrTelephoneTypeCodes[i];
        telephoneType.Name = arrTelephoneTypeNames[i];

        // Adicionar el tipo de telefono a la lista
        lstTelephoneTypes.Add(telephoneType);
    }

    // Retornar la lista de Tipos de telefonos
    return lstTelephoneTypes;
}

/// <summary>
/// Enlazar el campo Id con el combo
/// </summary>
/// <param name="textBox">Texto del Id</param>
/// <param name="comboBox">combo a enlazar</param>
private void BindPersonId(TextBox textBox, ComboBox comboBox)
{
    // Objeto para enlazar las propiedades
    Binding bnd = new Binding("Text", comboBox.DataSource, "Person_Id");
    // Enlazar el valor al control de texto
    textBox.DataBindings.Add(bnd);
}

```



```

/// <summary>
/// Cargar los datos de las personas
/// </summary>
/// <param name="cmb">Combo donde se asignan los datos</param>
public void LoadPersons(ComboBox cmb)
{
    // Instanciar logica de Negocio
    PersonBL personBL = new PersonBL();

    // Obtener la lista de las personas
    IList<PersonLight> lstPersonLight = personBL.ReadAllPersonsLight();

    // Validar si hay personas
    if (lstPersonLight.Count == 0)
    {
        // Mostrar el Mensaje de advertencia que no hay personas
        MessageBox.Show("Adicione primero registros de personas", "Advertencia",
        MessageBoxButtons.OK, MessageBoxIcon.Warning);

        // Deshabilitar los botones personalizados del Navigator
        this.bindingNavigatorAddNewItem.Enabled = false;
        this.bindingNavigatorDeleteItem.Enabled = false;
        this.bindingNavigatorSaveItems.Enabled = false;
    }

    // Asignar la lista a la fuente de datos del combo
    cmb.DataSource = lstPersonLight;

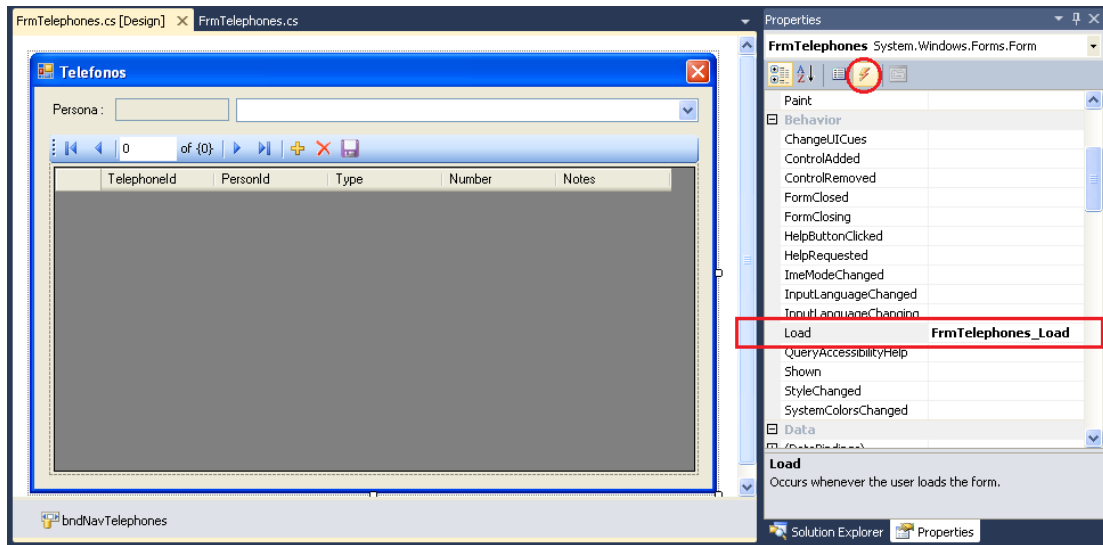
    // Asignar las propiedades del Combo
    cmb.DisplayMember = "FullName";
    cmb.ValueMember = "Person_Id";
}

#endregion Privados

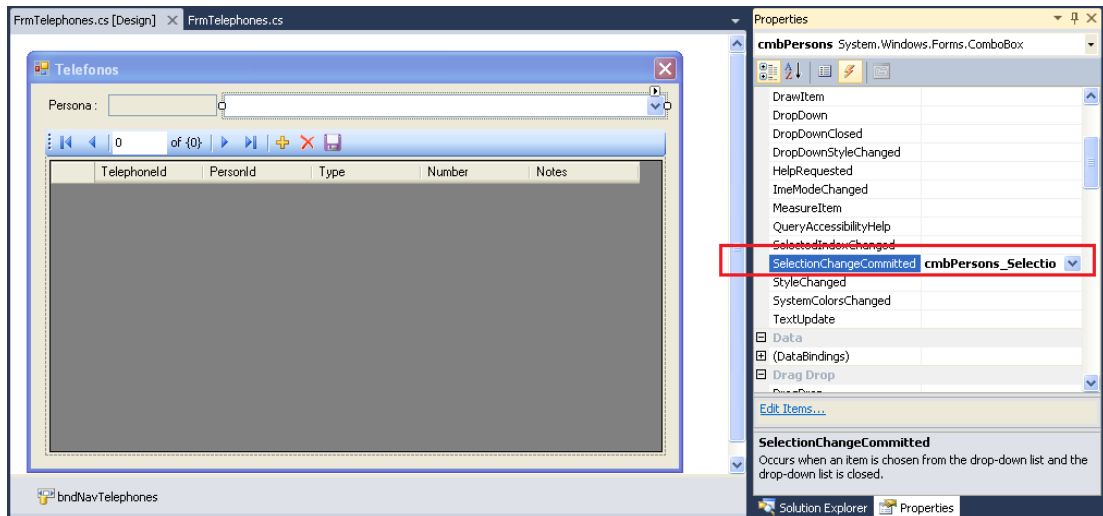
#endregion Metodos

```

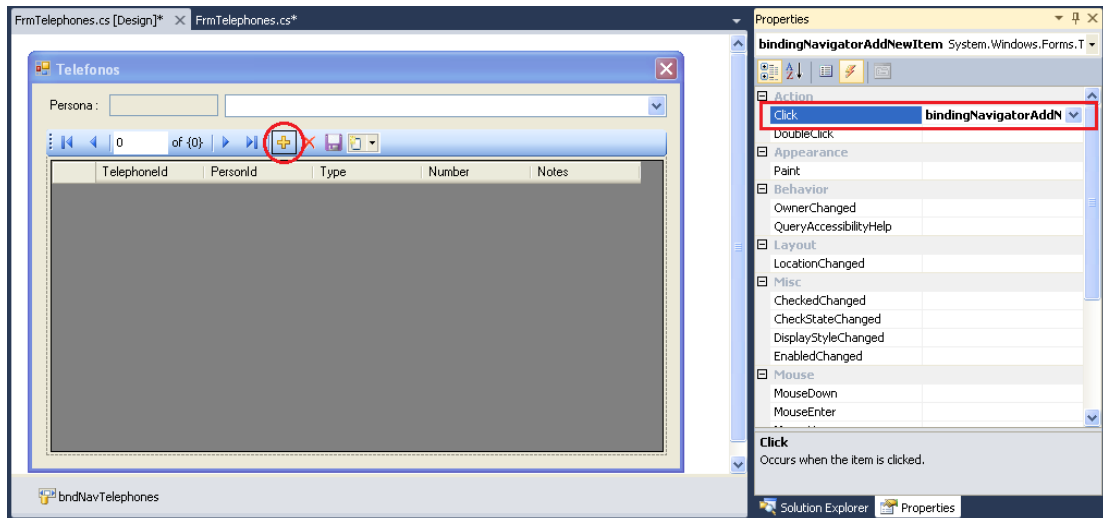
26. Seleccione la vista de diseño del formulario **FrmPerson**.
27. Seleccione las propiedades del formulario y active los eventos del mismo seleccionando el botón del trueno.



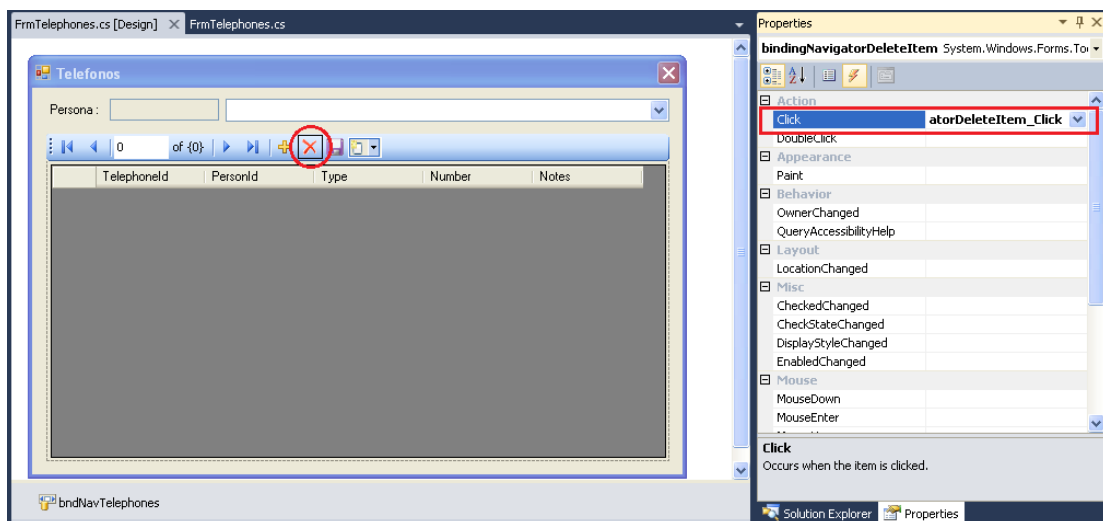
28. Seleccione el evento **Load** y escoja el valor **FrmTelephones_Load**.
29. Ahora seleccione el combo **cmbTelephones** y en el evento **SelectionChangeCommitted** escoja el valor **cmbPerson_SelectionChangeCommitted**.



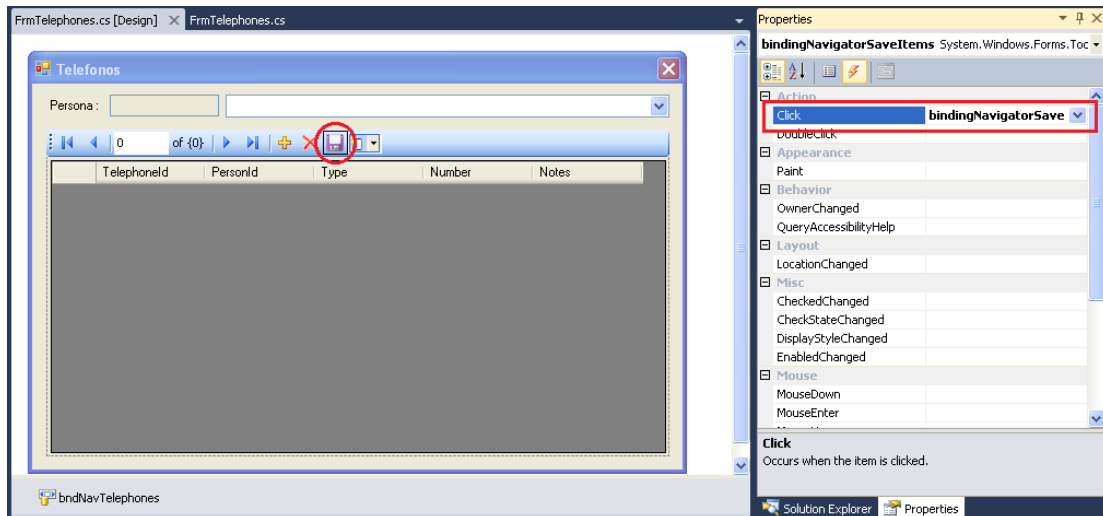
30. Ahora en el navegador enlazado (Binding Navigator) seleccione el botón de adicionar (**bindingNavigatorAddNewItem**) y en el evento **Click**, escoja el valor **bindingNavigatorAddNewItem_Click**.



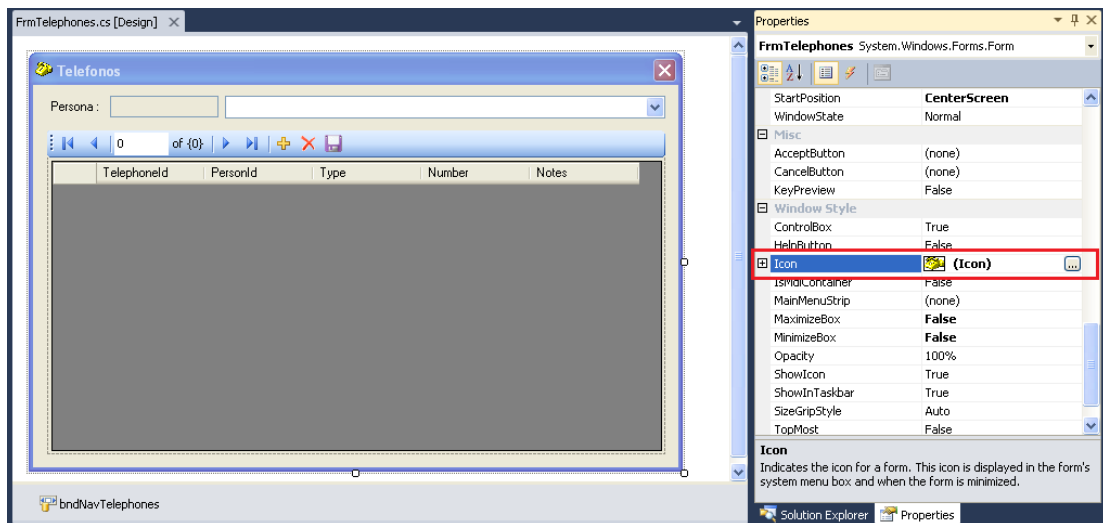
31. Ahora en el navegador enlazado (Binding Navigator) seleccione el botón de eliminar (**bindingNavigatorDeleteItem**) y en el evento **Click**, escoja el valor **bindingNavigatorDeleteItem_Click**.



32. Ahora en el navegador enlazado (Binding Navigator) seleccione el botón de Grabar (**bindingNavigatorSaveItems**) y en el evento **Click**, escoja el valor **bindingNavigatorSaveItems_Click**.



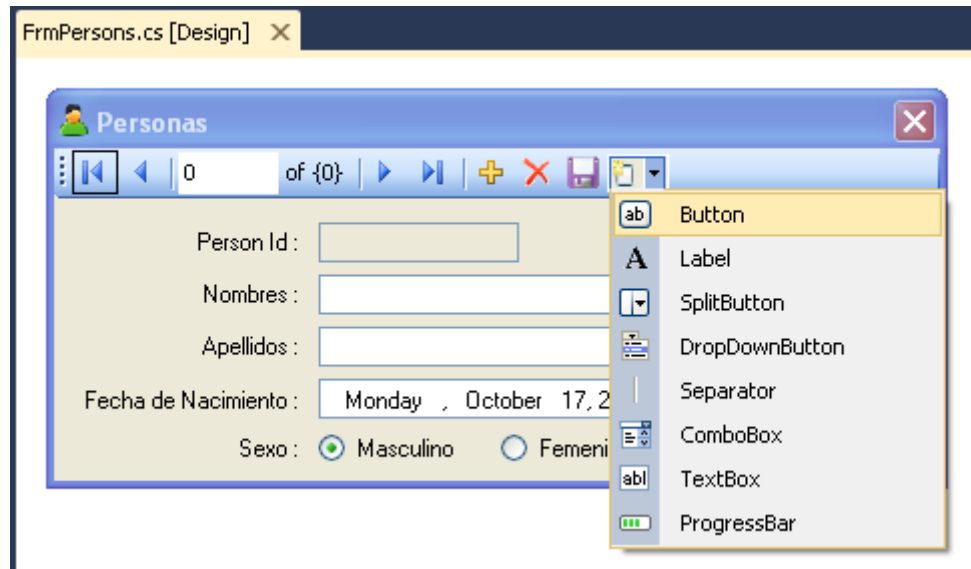
33. Seleccione el formulario y cambie el icono, importando el archivo **Phone06.ico** del directorio de recursos (**C:\Proyecto\SupportData\Recursos\lco**)



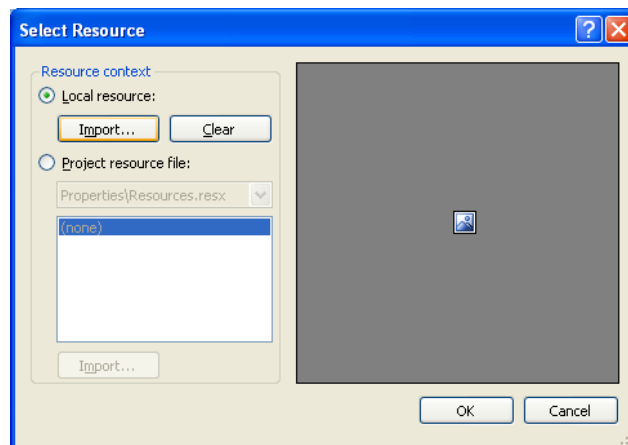
34. Guarde todos los cambios y compile el proyecto de interfaz de usuario.

FrmPersons

1. Seleccione el formulario de personas **FrmPersons** activando su vista de diseño.
2. Seleccione el Navegador enlazado de datos (Binding Navigator) **bndNavPersons** y adicione un nuevo botón.

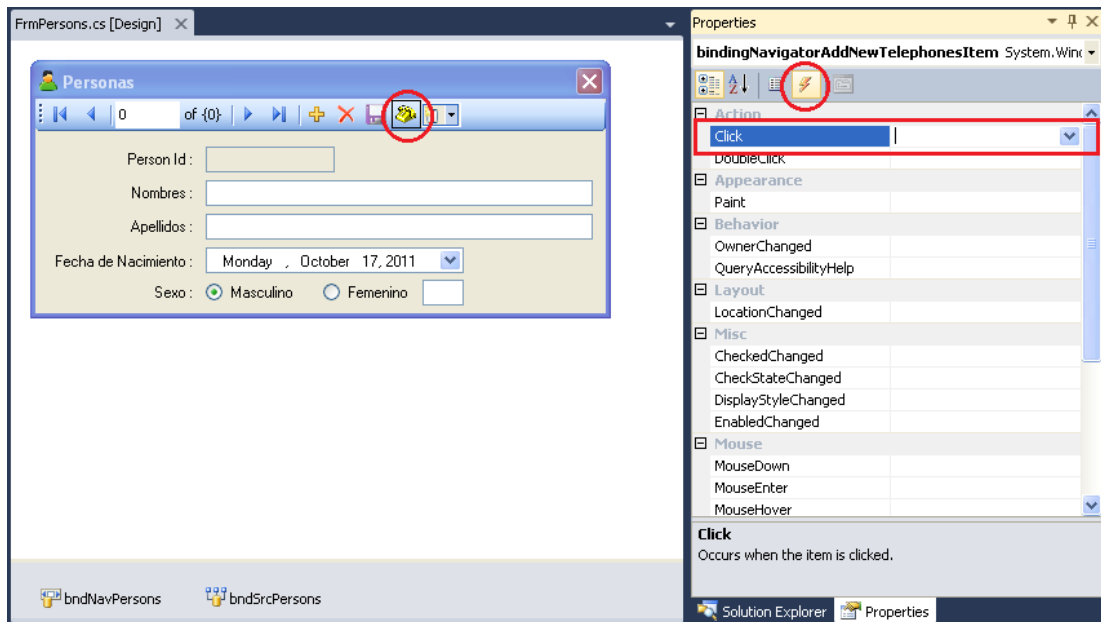


3. Cambiamos las propiedades de este nuevo botón de la siguiente forma:
 - a. **Name**= bindingNavigatorAddNewTelephonesItem
 - b. **Text**= AddNewTelephones
 - c. **ToolTipText**= Add New Telephones
4. Seleccionamos el nuevo botón y desde la ventana de propiedades escogemos la propiedad **Image**, para lo cual se despliega una ventana como la siguiente:

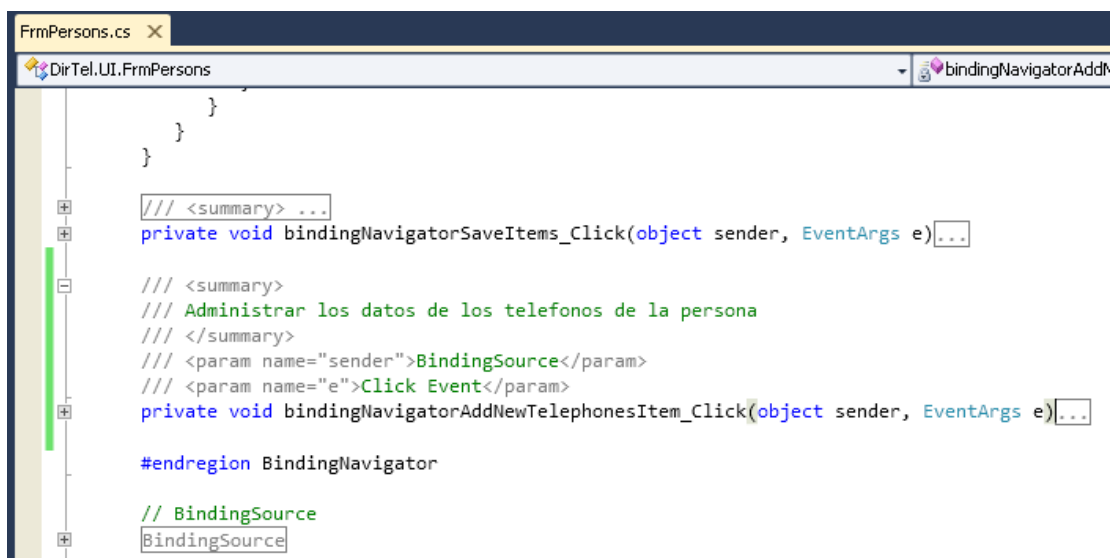


5. Escogemos la opción de importar del contexto local para que la imagen quede asociada al formulario y no al proyecto, esto nos facilita que al copiar el formulario a otra solución o proyecto este se vaya con la imagen incrustada dentro del formulario.

- Escogemos del directorio de **C:\Proyecto\SupportData\Recursos\lco**, la imagen de **Phone06.ico**, para ello debemos seleccionar la lista de todos los archivo (All Files *.*) y aceptamos el nuevo recurso
- Seleccione el botón y active en la ventana de propiedades la vista de eventos, escogiendo el botón del **trueno** y en el evento **Click**, haga doble click sobre el mismo.



- Esta última acción agrego el método **bindingNavigatorAddNewTelephonesItem_Click**.
- En el editor de texto mueva este nuevo método (**bindingNavigatorAddNewTelephonesItem_Click**) a la región de BindingNavigator, después del método **bindingNavigatorSaveItems_Click** y modifíquelo de la siguiente forma:



10. Edite el contenido del método de la siguiente forma:

```
/// <summary>
/// Administrar los datos de los telefonos de la persona
/// </summary>
/// <param name="sender">BindingSource</param>
/// <param name="e">Click Event</param>
private void bindingNavigatorAddNewTelephonesItem_Click(object sender, EventArgs e)
{
    // Poner el Foco en el campo del Id
    txtPersonId.Focus();

    // Validar si el Id de la persona no es vacio
    if (!txtPersonId.Text.Trim().Equals(string.Empty))
    {
        // Obtener el Id de la persona actual
        long person_Id = Convert.ToInt64(this.txtPersonId.Text);

        // Instanciar un nuevo formulario de Telefonos
        FrmTelephones frmTelephones = new FrmTelephones();

        // Asignar el Identificador de la persona al formulario
        frmTelephones.Person_Id = person_Id;

        // Asignar el formulario padre
        frmTelephones.MdiParent = this.MdiParent;

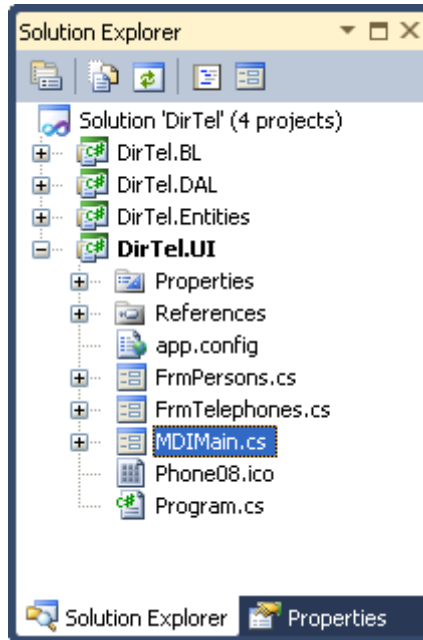
        // Mostrar el formulario
        frmTelephones.Show();
    }
}
```

11. Guarde los cambios y compile el proyecto de interfaz de usuario.

MDIMain

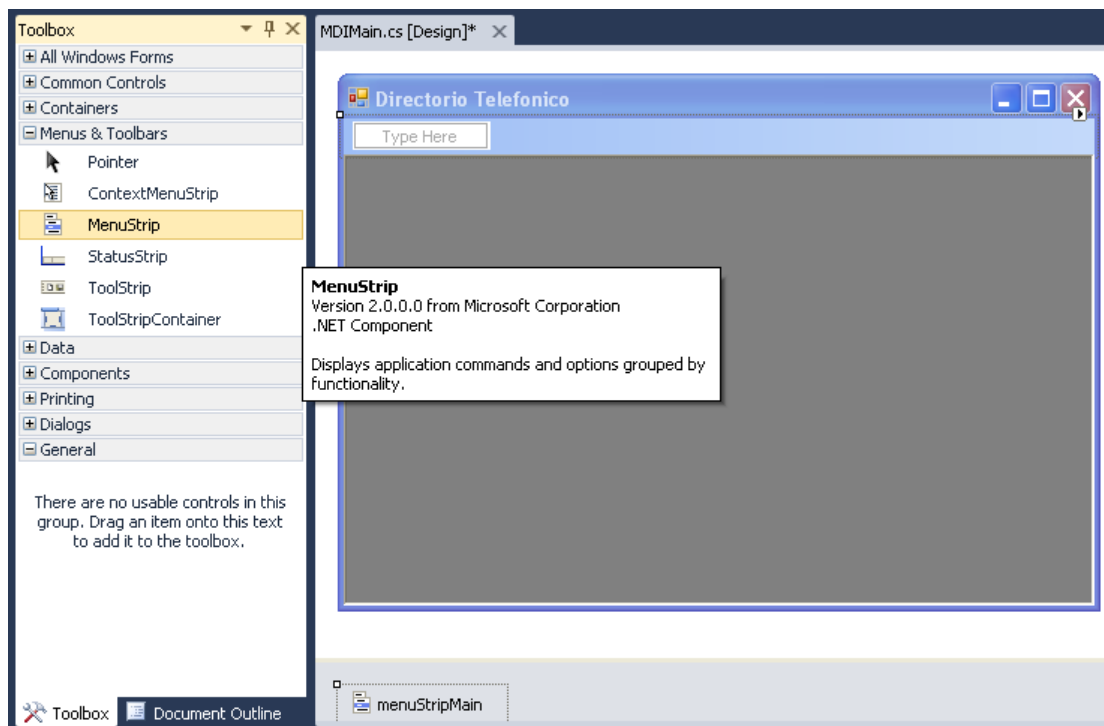
Ahora vamos a completar nuestra aplicación creando un formulario que contenga el menú para llamar a los otros formularios. Este tipo de formularios se denomina MDI (Multiple Document Interface)

1. Adicione al proyecto de interfaz de usuario (DirTel.UI) un nuevo formulario, denominado **MDIMain**. Para este ejercicio no vamos a utilizar la opción de MDI form sino la de Windows Form para crear un formulario MDI de forma manual.

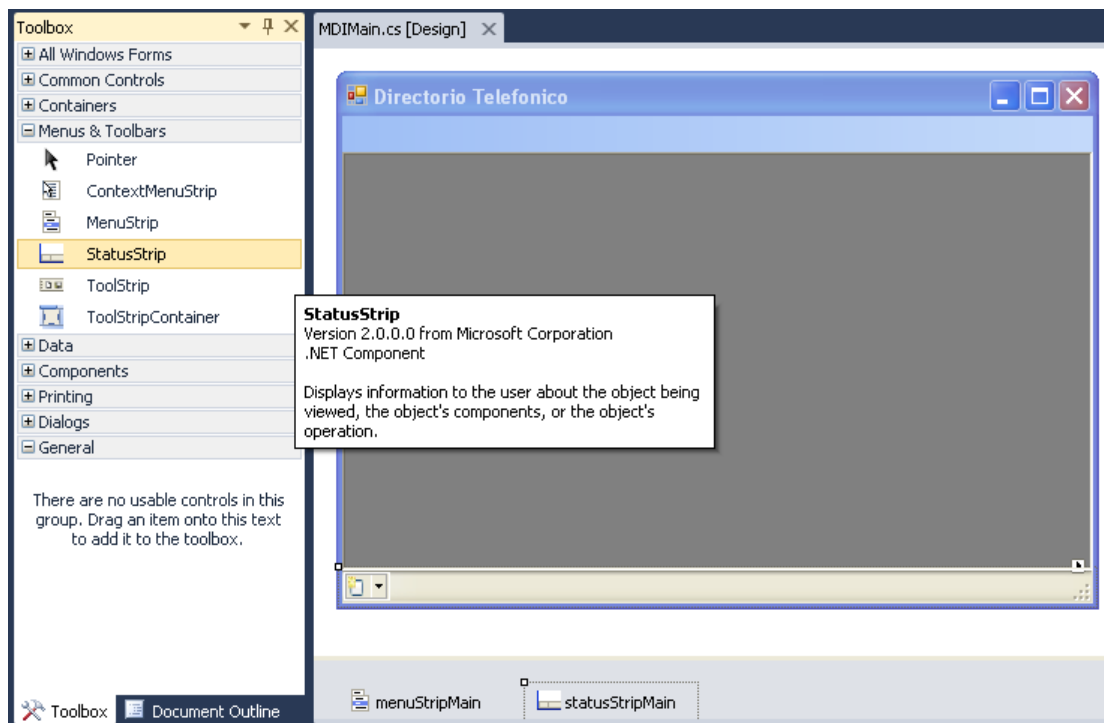


2. Ahora vamos a modificar las propiedades de este nuevo formulario para establecerlo como un MDI Form.
 - a. **IsMdiContainer** = True
 - b. **StartPosition** = Center Screen
 - c. **Text** = Directorio Telefonico
 - d. **WindowState** = Maximized

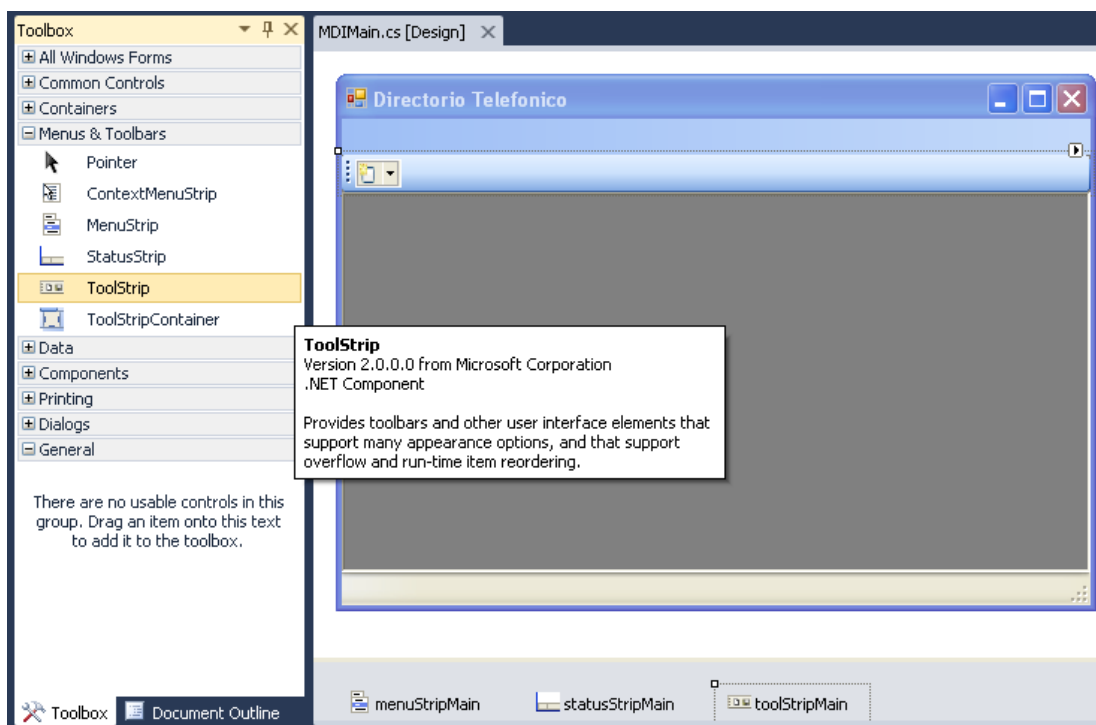
3. Adicione un control de **MenuStrip** al formulario y cambie su nombre a **menuStripMain**.



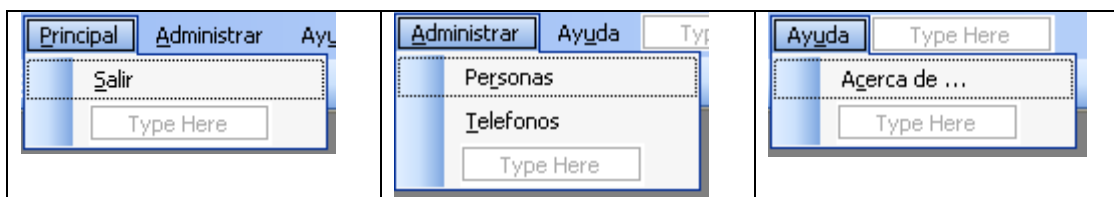
4. Adicione ahora un control de tipo **StatusStrip** y denomínelo **statusStripMain**.



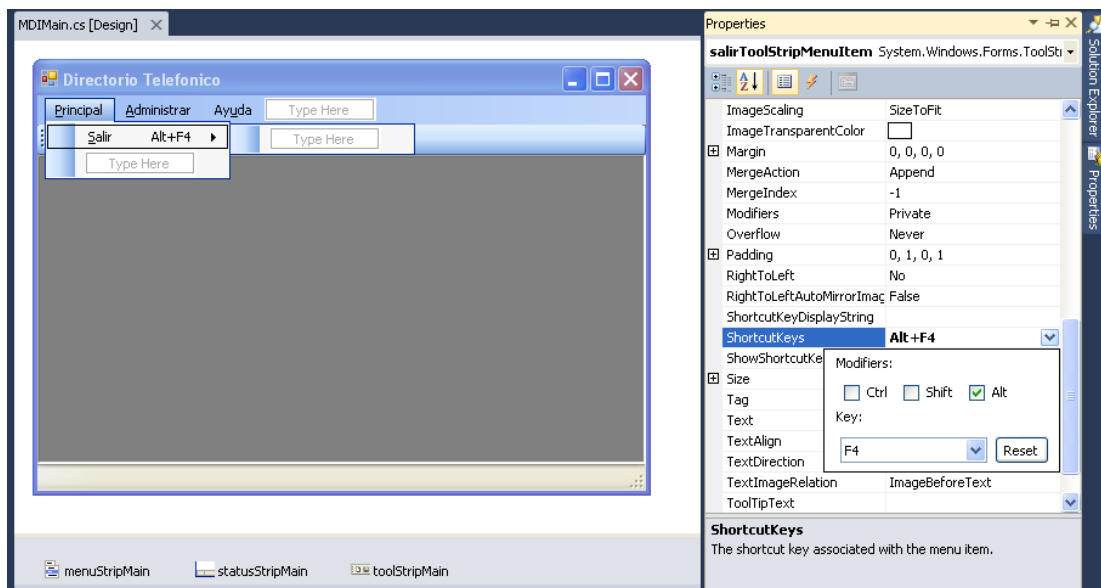
- Ahora adicione un control de tipo **ToolStrip** y denomínelo **toolStripMain**.



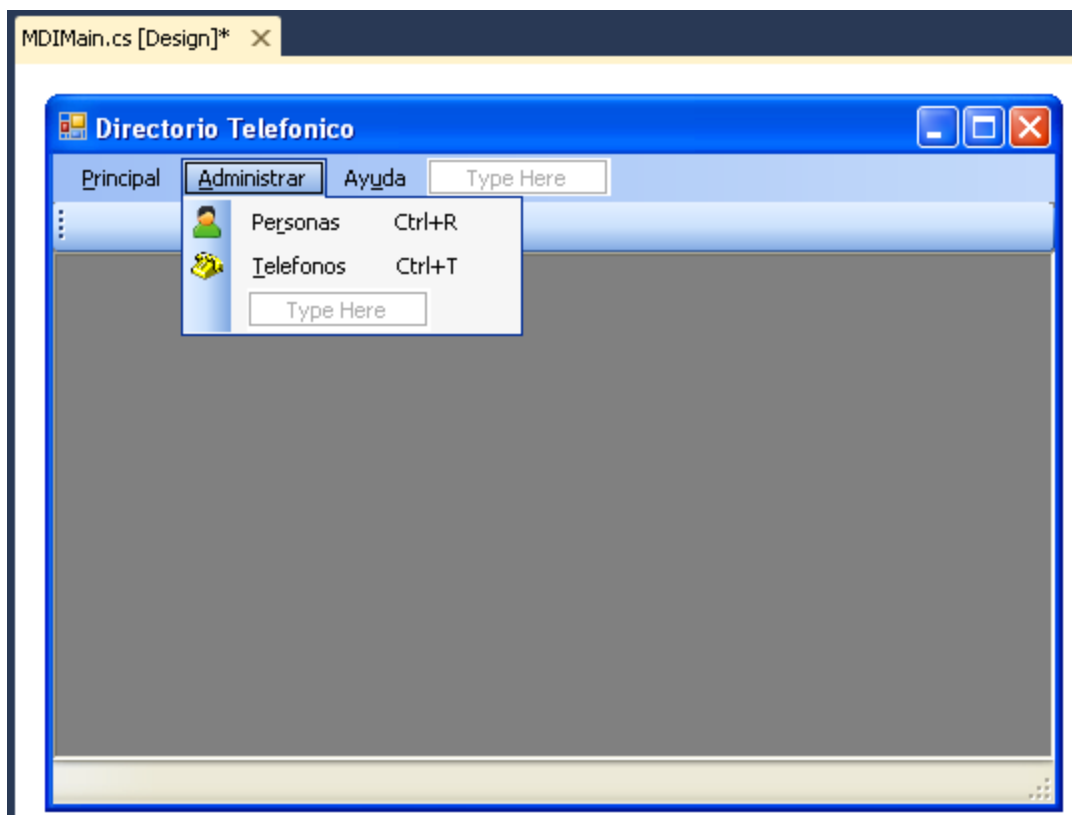
- Seleccione el Menú y construya la siguiente estructura.



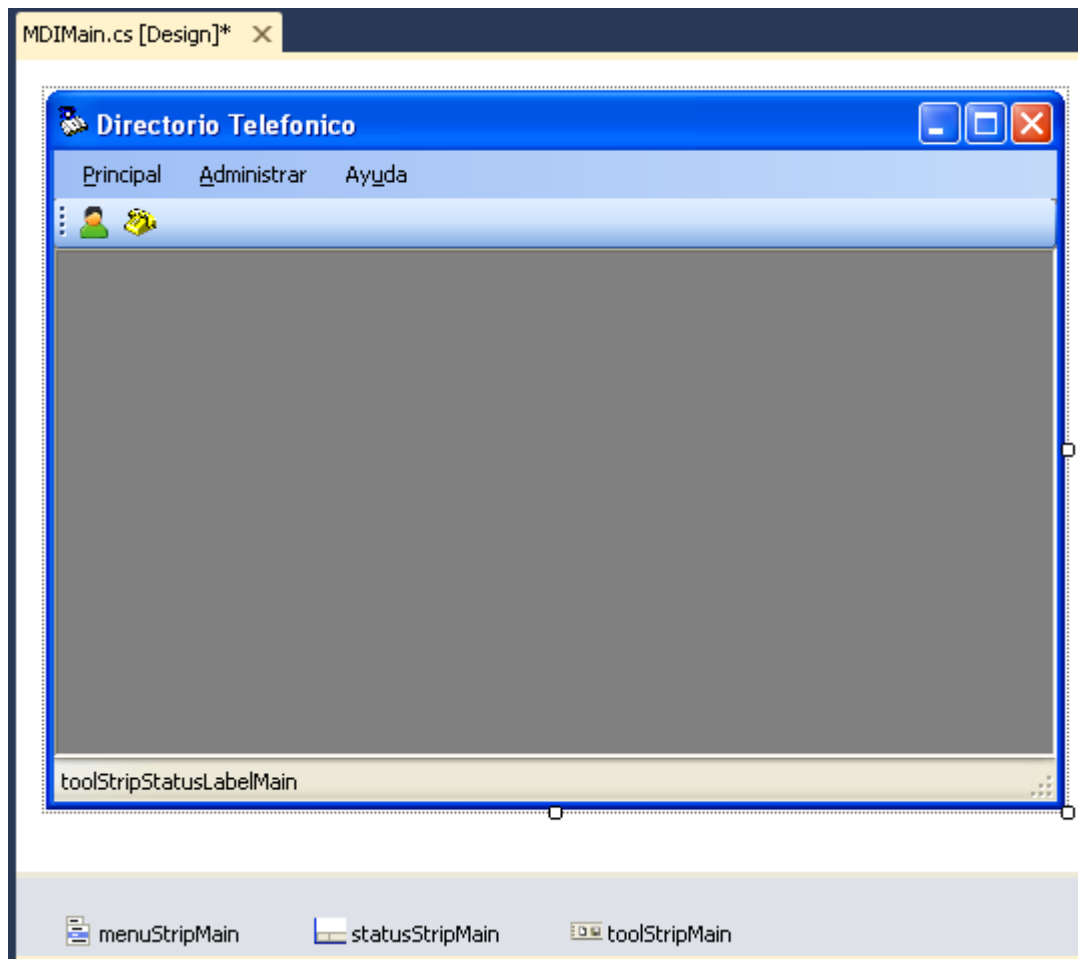
- Para poner un acceso rápido en cualquiera de las letras de las opciones puede anteponer el símbolo Ampersand (&) antes de la letra a seleccionar, por ejemplo para seleccionar la letra **P** en la palabra principal debemos escribir &Principal lo que nos da como resultado Pincipal. Realice esta operación para cada una de las letras subrayadas del menú.
- Ahora si desea adicionar un conjunto de teclas rápidas para cada opción puede escogerla del menú y en la ventana de propiedades seleccionar la propiedad **ShortcutKeys**.



9. Asigne las siguientes teclas rápidas para las opciones del Menú.
 - a. Salir = CTRL+F4
 - b. Personas = CTRL+R
 - c. Telefonos = CTRL+F
10. Ahora modifique las opciones del Menú de **Personas** y de **Telefonos** y asigne los iconos de **Contacts08.ico** y **Phone06.ico** respectivamente.

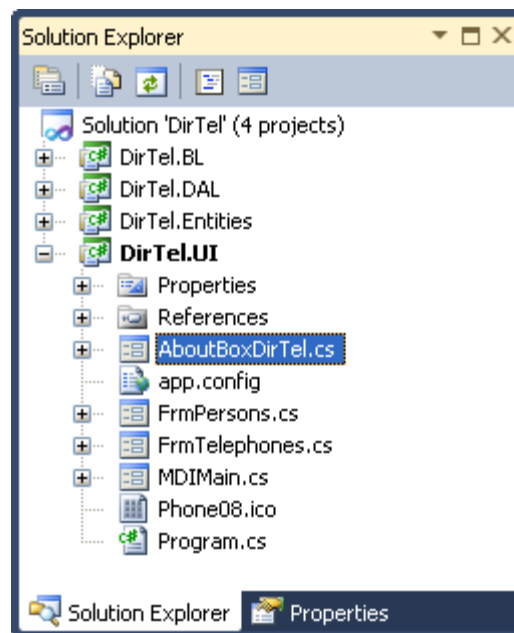
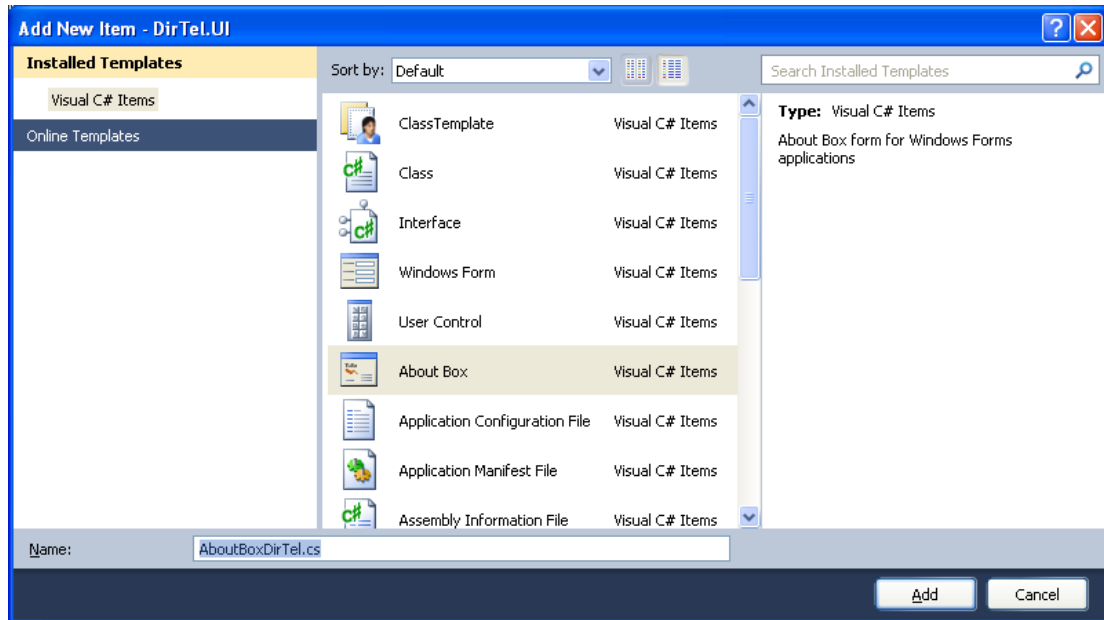


11. Seleccione el control de **ToolStripMain** y adicione dos botones:
 - a. **toolStripButtonPersons**
 - b. **toolStripButtonTelephones**
12. Modifique sus propiedades así:
 - a. **toolStripButtonPersons**
 - i. **Image** = C:\Proyecto\SupportData\Recursos\lco\Contacs08.ico
 - ii. **Text** = **toolStripButtonPersons**
 - iii. **ToolTipText** = Personas
 - b. **toolStripButtonTelephones**
 - i. **Image** = C:\Proyecto\SupportData\Recursos\lco\Phone06.ico
 - ii. **Text** = **toolStripButtonTelephones**
 - iii. **ToolTipText** = Telefonos
13. Seleccione el control **StatusStripMain** y adicione un control **StatusLabel** denominado **toolStripStatusLabelMain**.
14. Seleccione el formulario principal y cambie su icono por el **Phone08.ico**, del directorio de **C:\Proyecto\SupportData\Recursos\lco**.



Acerca de ...

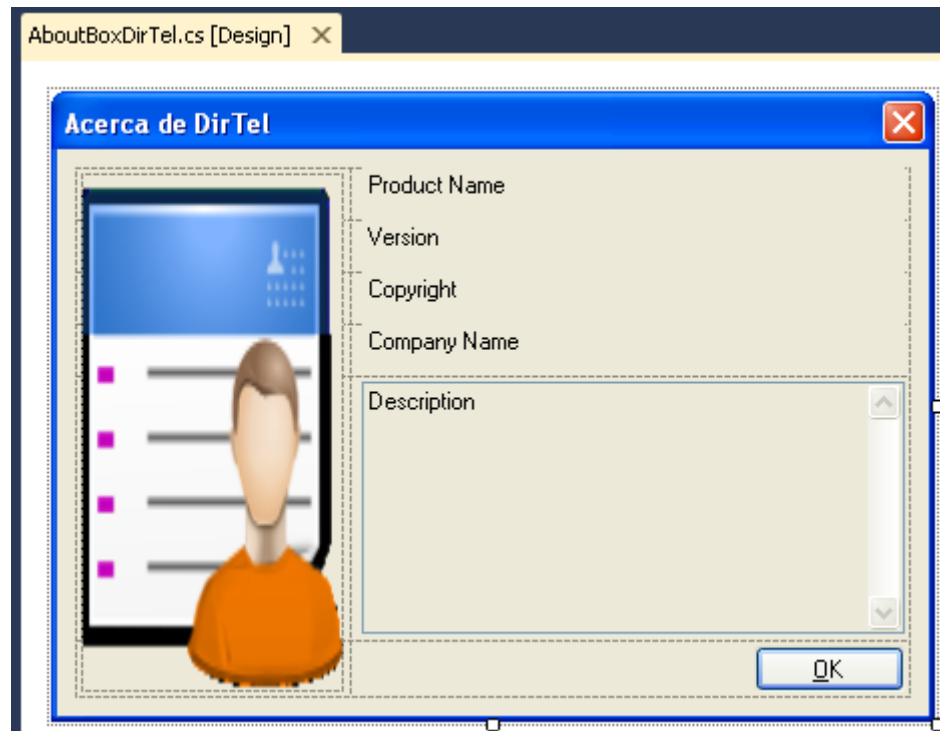
15. Adicione un nuevo formulario de tipo **AboutBox** al proyecto de interfaz de usuario denominado **AboutBoxDirTel.cs**.



16. Esto agrega el siguiente formulario.

The image shows a Windows Forms application in design mode. The title bar of the IDE window is 'AboutBoxDirTel.cs [Design]'. The form being designed is titled 'AboutBoxDirTel'. On the left side of the form is a graphic of three interlocking rings in green, orange, and blue. To the right of the graphic are four text labels: 'Product Name', 'Version', 'Copyright', and 'Company Name'. Below these labels is a large text area labeled 'Description' with a vertical scrollbar. An 'OK' button is located at the bottom right of the form.

17. Modifique la imagen y seleccione el icono **Contacts07.ico** del directorio de **C:\Proyecto\SupporData\Recursos\lco**.
18. Modifique las propiedades del formulario así:
 - a. **Text** = Acerca de DirTel
 - b. **StartPosition** = Center Screen

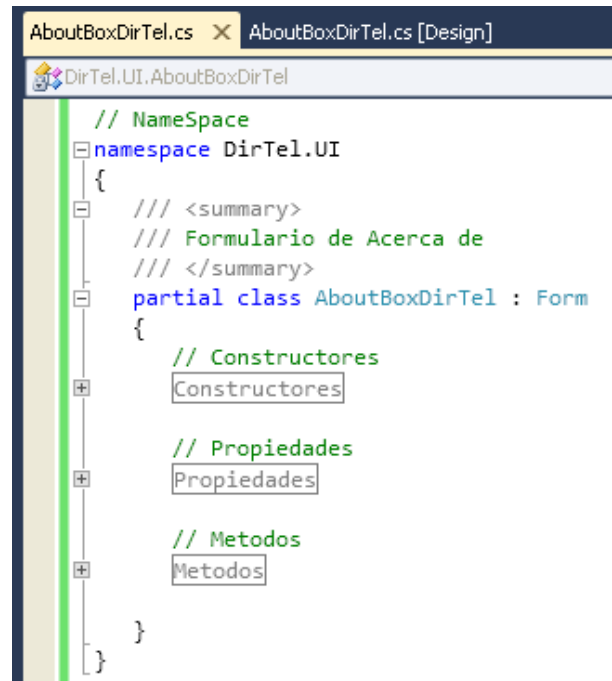


19. Active el código del formulario presionando la tecla **F7**.
20. Edite el contenido del archivo **AboutBoxdirTel.cs** y adicione la cabecera de comentarios correspondiente.

```

AboutBoxDirTel.cs  AboutBoxDirTel.cs [Design]
DirTel.UI, AboutBoxDirTel  #AboutBoxDirTel()
#region Documentación
/*****
 * Propiedad intelectual de Engineers & Tools (c).
 *****/
 * Unidad      : AboutBoxDirTel.cs
 * Descripcion : Formulario Acerca de .. de la aplicacion
 * Autor       : Julio Cesar Robles Uribe - Jucer
 * Fecha       : 29-May-2010
 *
 * Fecha      Autor      Modificación
 * =====
 * 29-May-2010 Jucer      1 - Version Inicial
 *****/
#endregion Documentación
  
```

21. Modifique la documentación de la clase.

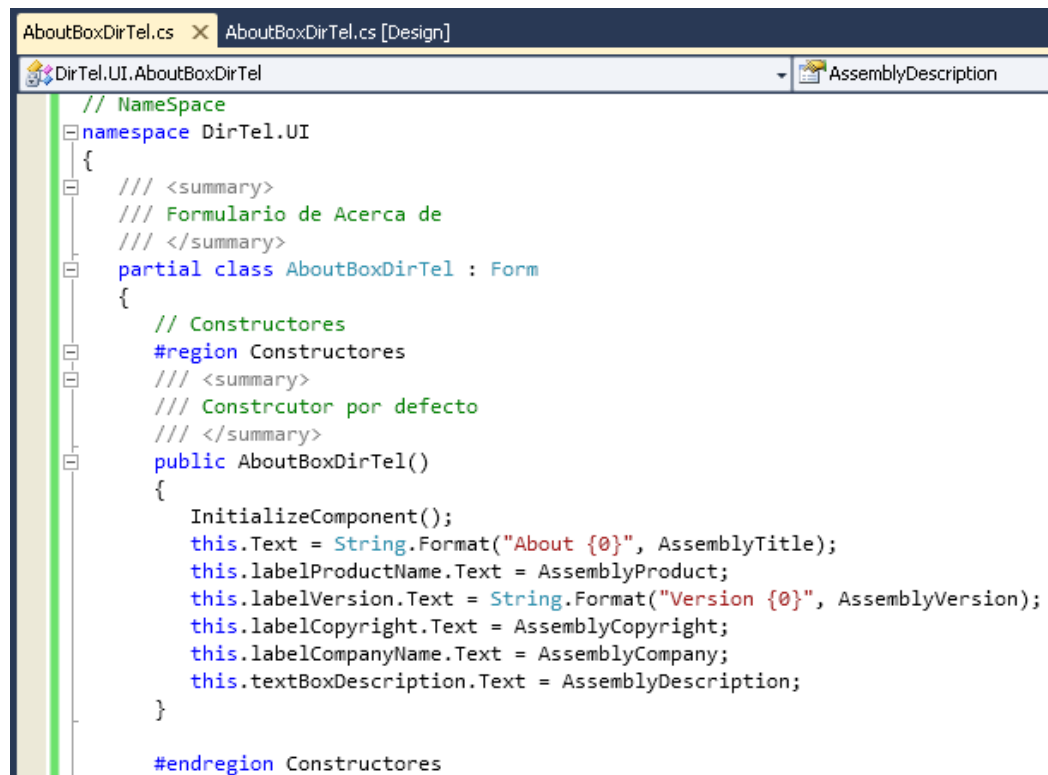


```
// NameSpace
namespace DirTel.UI
{
    /// <summary>
    /// Formulario de Acerca de
    /// </summary>
    partial class AboutBoxDirTel : Form
    {
        // Constructores
        Constructores

        // Propiedades
        Propiedades

        // Metodos
        Metodos
    }
}
```

22. Documente y regionalice el constructor.



```
// NameSpace
namespace DirTel.UI
{
    /// <summary>
    /// Formulario de Acerca de
    /// </summary>
    partial class AboutBoxDirTel : Form
    {
        // Constructores
        #region Constructores
        /// <summary>
        /// Constructor por defecto
        /// </summary>
        public AboutBoxDirTel()
        {
            InitializeComponent();
            this.Text = String.Format("About {0}", AssemblyTitle);
            this.labelProductName.Text = AssemblyProduct;
            this.labelVersion.Text = String.Format("Version {0}", AssemblyVersion);
            this.labelCopyright.Text = AssemblyCopyright;
            this.labelCompanyName.Text = AssemblyCompany;
            this.textBoxDescription.Text = AssemblyDescription;
        }
        #endregion Constructores
    }
}
```


23. Documente y regionalice las propiedades

```
// Namespace
namespace DirTel.UI
{
    /// <summary>
    /// Formulario de Acerca de
    /// </summary>
    partial class AboutBoxDirTel : Form
    {
        // Constructores
        Constructores

        // Propiedades
        #region Propiedades

        // Obtener los datos del Assembly
        #region Assembly Attribute Accessors

        /// <summary>
        /// Obtener el Titulo del Assembly
        /// </summary>
        public string AssemblyTitle...

        /// <summary>
        /// Obtener la version del Assembly
        /// </summary>
        public string AssemblyVersion...

        /// <summary>
        /// Obtener la descripcion del Assembly
        /// </summary>
        public string AssemblyDescription...

        /// <summary>
        /// Obtener el Producto
        /// </summary>
        public string AssemblyProduct...

        /// <summary>
        /// Obtner los derechos de autor
        /// </summary>
        public string AssemblyCopyright...

        /// <summary>
        /// Obtener la Empresa
        /// </summary>
        public string AssemblyCompany...

        #endregion Assembly Attribute Accessors

        #endregion Propiedades

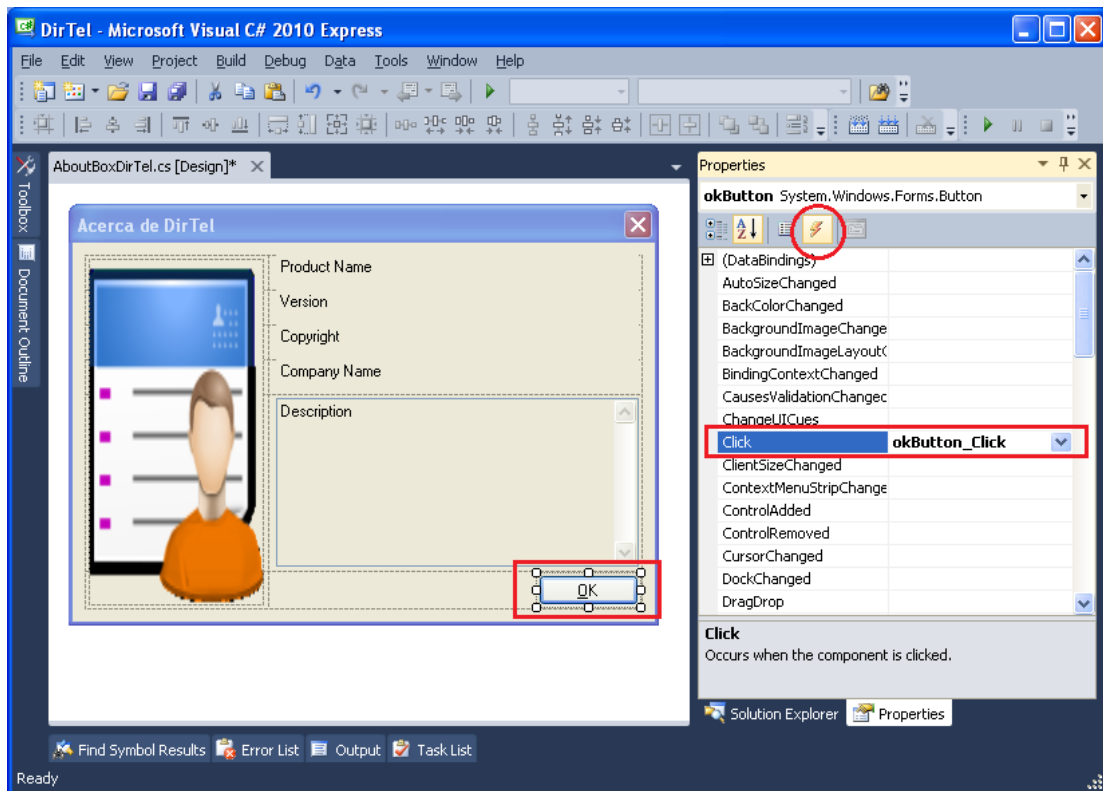
        // Metodos
        Metodos
    }
}
```

24. Adicione la región de Métodos.

```
// Metodos
#region Metodos

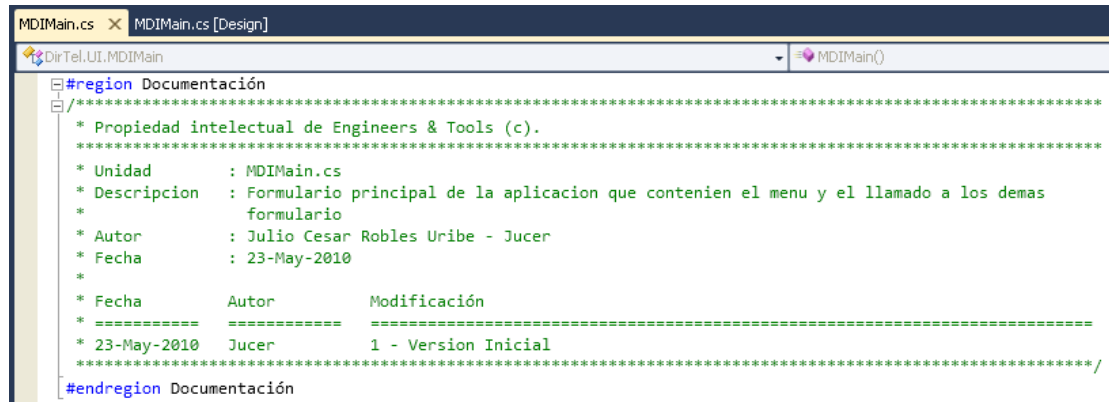
/// <summary>
/// Aceptar la informacion y cerrar el formulario
/// </summary>
/// <param name="sender">Boton Ok</param>
/// <param name="e">Click</param>
private void okButton_Click(object sender, EventArgs e)
{
    // Cerrar el formulario
    this.Close();
}
#endregion Metodos
```

25. Active la vista de diseño del formulario **AboutBoxDirTel** y seleccione el botón **OK** . De la ventana de propiedades active la vista de Eventos, mediante el botón del trueno y escoja el evento Click y asigne el valor de `okButton_Click`.



Código MDIMain

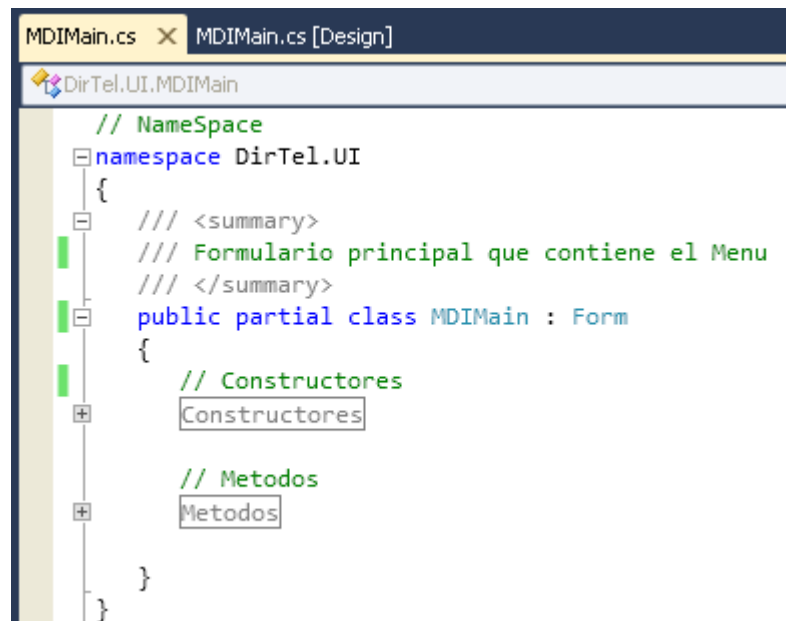
26. Edite el contenido del archivo **MDIMain.cs** y modifique el encabezado con los datos correspondientes.



```
MDIMain.cs x MDIMain.cs [Design]
DirTel.UI.MDIMain
MDIMain()

#region Documentación
/*****
 * Propiedad intelectual de Engineers & Tools (c).
 *****/
 * Unidad      : MDIMain.cs
 * Descripción : Formulario principal de la aplicación que contienen el menú y el llamado a los demás formularios
 * Autor       : Julio Cesar Robles Uribe - Jucer
 * Fecha       : 23-May-2010
 *
 * Fecha      Autor      Modificación
 * =====
 * 23-May-2010 Jucer      1 - Version Inicial
 *****/
#endregion Documentación
```

27. Modifique el alcance de la clase **MDIMain** a **public** y actualice la documentación correspondiente.

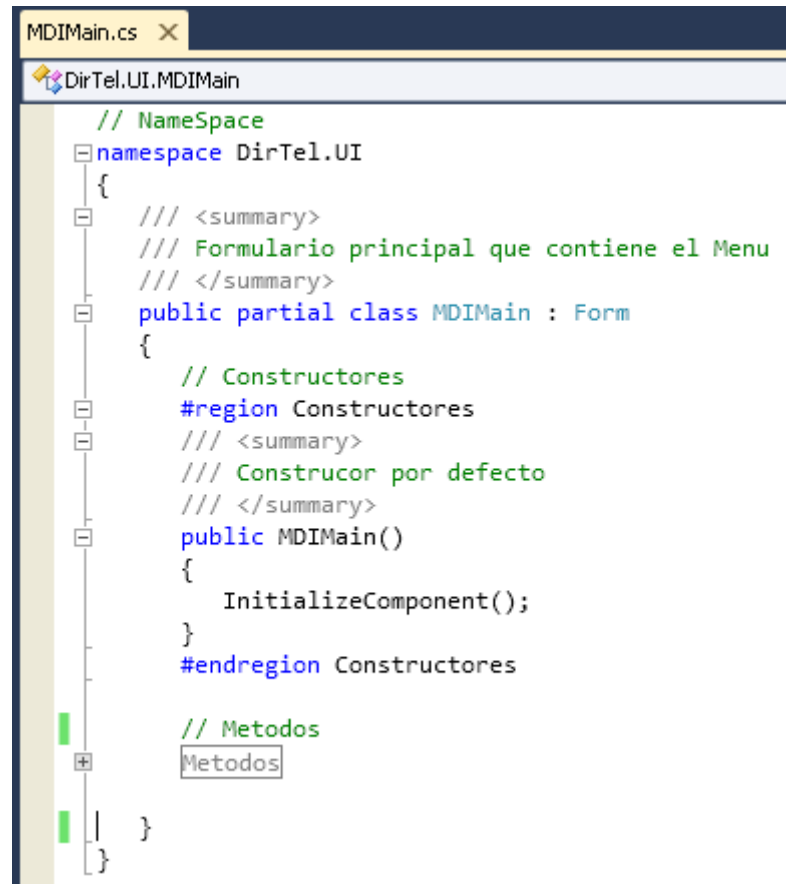


```
MDIMain.cs x MDIMain.cs [Design]
DirTel.UI.MDIMain

// Namespace
namespace DirTel.UI
{
    /// <summary>
    /// Formulario principal que contiene el Menu
    /// </summary>
    public partial class MDIMain : Form
    {
        // Constructores
        Constructores

        // Metodos
        Metodos
    }
}
```

28. Documente el constructor por defecto



```
MDIMain.cs X
DirTel.UI.MDIMain

// Namespace
namespace DirTel.UI
{
    /// <summary>
    /// Formulario principal que contiene el Menu
    /// </summary>
    public partial class MDIMain : Form
    {
        // Constructores
        #region Constructores
        /// <summary>
        /// Construtor por defecto
        /// </summary>
        public MDIMain()
        {
            InitializeComponent();
        }
        #endregion Constructores

        // Metodos
        Metodos
    }
}
```

29. Adicione los métodos

```
// Metodos
#region Metodos

// Privados
#region Privados

/// <summary>
/// Salir de la aplicacion
/// </summary>
/// <param name="sender">Menu Salir</param>
/// <param name="e">Click</param>
private void salirToolStripMenuItem_Click(object sender, EventArgs e)
{
    // Cerrar el formulario principal
    this.Close();

    // Salir de la aplicacion
    Application.Exit();
}
```

```

/// <summary>
/// Administrar Personas
/// </summary>
/// <param name="sender">Opcion Personas</param>
/// <param name="e">Click</param>
private void personasToolStripMenuItem_Click(object sender, EventArgs e)
{
    // Instanciar un formulario de Personas
    FrmPersons frmPersons = new FrmPersons();

    // Asignar el formulario padre
    frmPersons.MdiParent = this;

    // Mostrar el Formulario
    frmPersons.Show();
}

/// <summary>
/// Administrar Telefonos
/// </summary>
/// <param name="sender">Opcion Telefonos</param>
/// <param name="e">Click</param>
private void telefonosToolStripMenuItem_Click(object sender, EventArgs e)
{
    // Instanciar un formulario de Telefonos
    FrmTelephones frmTelephones = new FrmTelephones();

    // Asignar el formulario padre
    frmTelephones.MdiParent = this;

    // Mostrar el Formulario
    frmTelephones.Show();
}

/// <summary>
/// Ventana de Acerca de.. de la aplicacion
/// </summary>
/// <param name="sender">Menu Acerca de</param>
/// <param name="e">Click</param>
private void acercaDeToolStripMenuItem_Click(object sender, EventArgs e)
{
    // Instanciar el formulario de acerca de...
    AboutBoxDirTel about = new AboutBoxDirTel();

    // Asignar la ventana padre
    about.MdiParent = this;

    // Mostrar el formulario
    about.Show();
}

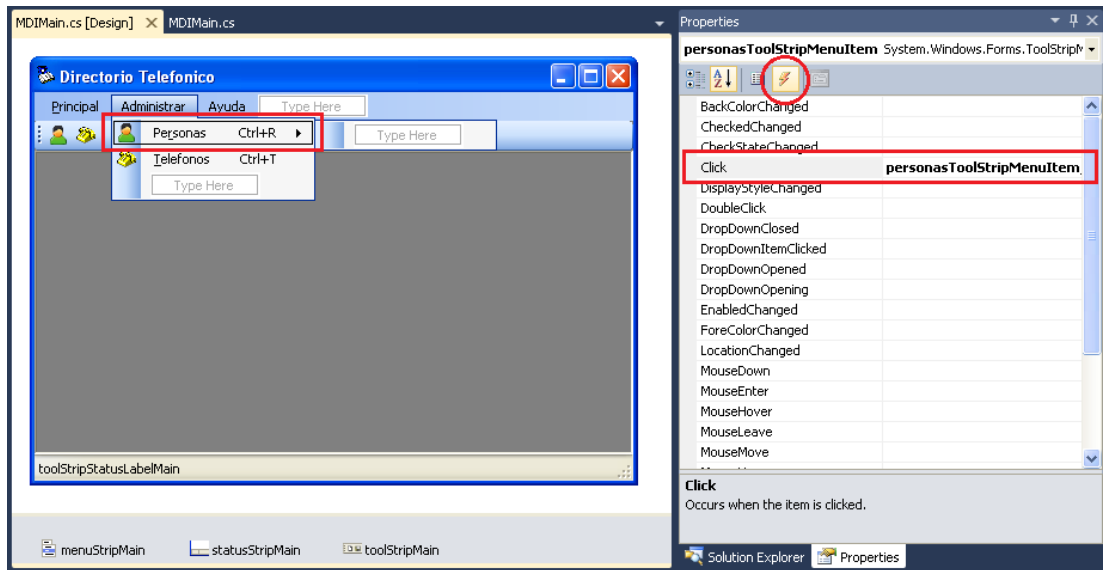
#endregion Privados

#endregion Metodos

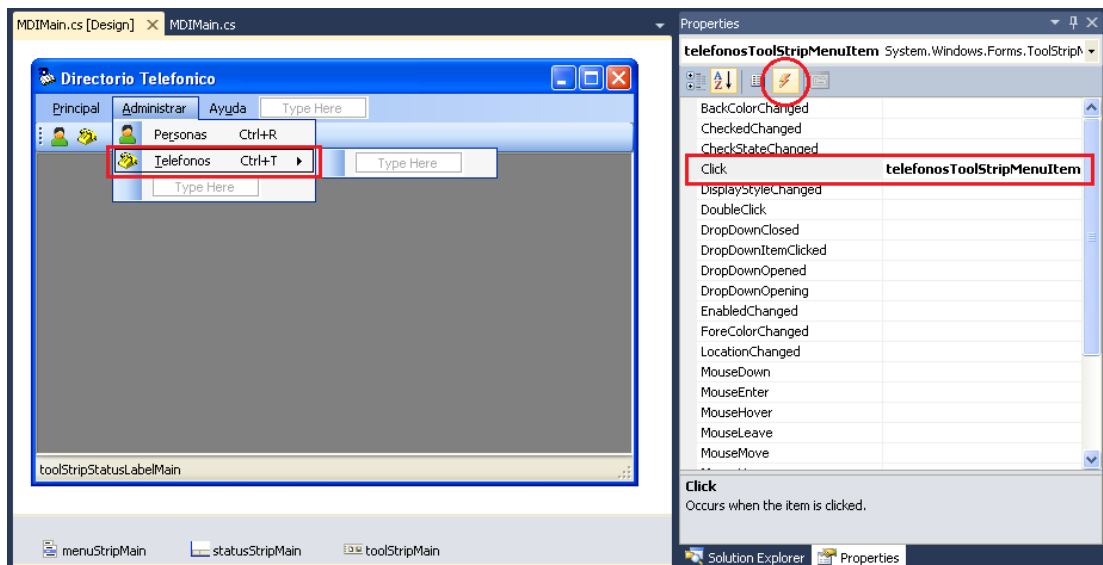
```

30. Vuelva a la vista de diseño del formulario MDIMain y seleccione el menú de Administrar.

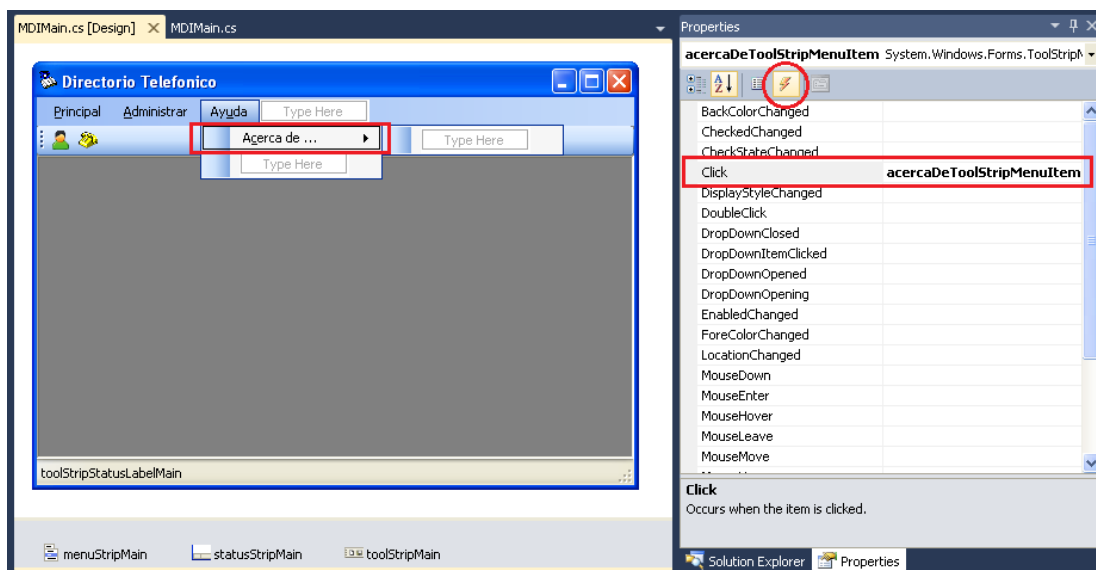
31. Seleccione la opción **Personas** y en la ventana de propiedades active la vista de eventos mediante el botón del trueno. De esta vista seleccione el evento **Click** y asigne el valor de **personasToolStripMenuItem_Click**.



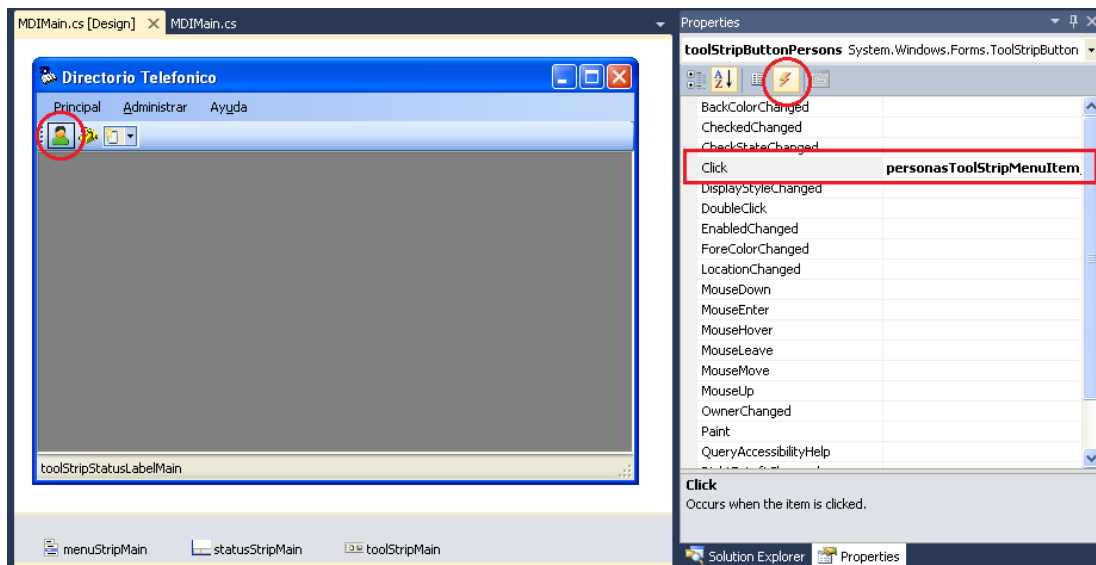
32. Ahora seleccione la opción **Telefonos** y en la ventana de propiedades active la vista de eventos mediante el botón del trueno. De esta vista seleccione el evento **Click** y asigne el valor de **telefonosToolStripMenuItem_Click**.



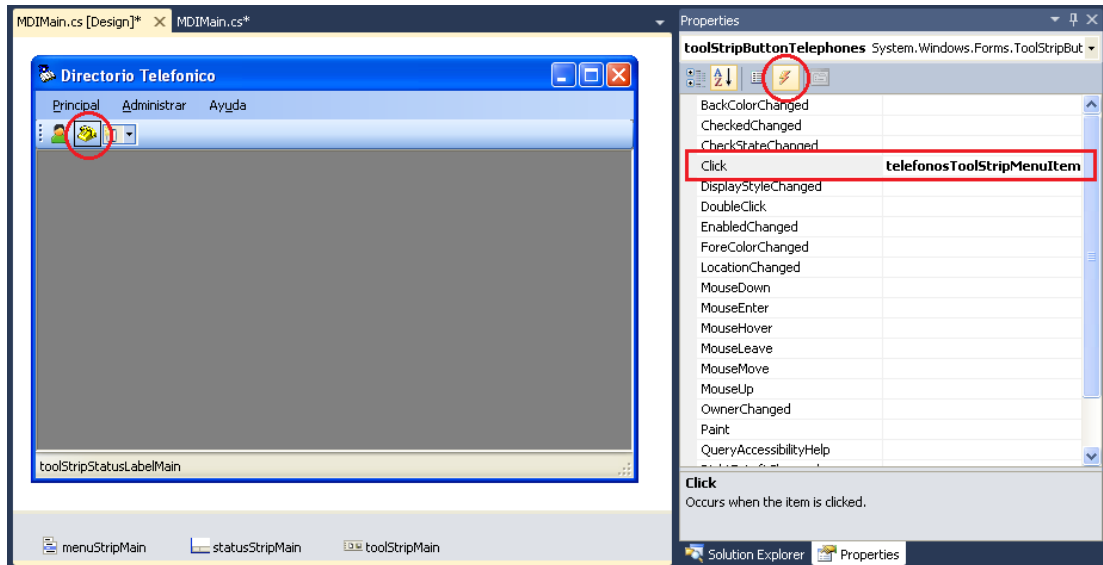
33. De igual forma seleccione la opción de Acerca de... del menú de Ayuda y de la ventana de propiedades activada la vista de eventos asigne al event Click el valor de `acercaDeToolStripMenuItem_Click`.



34. Ahora seleccione, del formulario principal, el botón de **Personas** del área de **ToolStripMain** y de la ventana de propiedades, teniendo activa la vista de eventos, seleccione el evento **Click** y asigne el valor de `personasToolStripMenuItem_Click`.



35. Luego seleccione, del formulario principal, el botón de **Telefonos** del área de **ToolStripMain** y de la ventana de propiedades, teniendo activa la vista de eventos, seleccione el evento **Click** y asigne el valor de **telefonosToolStripMenuItem_Click**.



36. Por último guarde los cambios y compile toda la solución.
37. Corrija los posibles errores y pruebe su funcionamiento.