**PROYECTO FINAL - TIENDA**

**LENGUAJE DE PROGRAMACION (ASSOCIATE)**

Se debe crear un sistema de información de escritorio para la Tienda “La Casita”; donde cumpla con las siguientes especificaciones técnicas y funcionales:

Especificaciones Técnicas y/o no Funcionales

1. Se debe implementar en lenguaje de programación JAVA o .NET C#
2. El motor de base de datos debe ser SQLServer Express o MySQL
3. La arquitectura debe ser en Capas (MVC)
4. Se debe diseñar el modelo relacional de la base de datos
5. Se debe realizar el diagrama de clases de la solución
6. El sistema debe contar con seguridad en el acceso por perfil (administrador y vendedor)
7. El sistema debe tener ayudas de uso en los formularios; debe ser intuitivo

Requerimientos Funcionales

El sistema debe contar con las siguientes condiciones y características basadas en la operación de la Tienda:

1. Debe almacenar la información básica de los clientes (Nro. Documento, Nombre, Dirección, Teléfono)
2. La tienda tiene organizados sus productos por categorías, donde las categorías cuentan con un Código y Nombre; y los productos tienen Nombre, marca, precio, stock
3. Los clientes pueden realizar pedidos, de los cuales es importante almacenar la fecha del pedido, el cliente, que usuario realiza la venta y el estado del Pedido (solicitado, despachado); los pedidos tienen un detalle del pedido donde se la cantidad y el valor total por producto.
4. Debe existir un almacenamiento de los usuarios del sistema donde estén los datos básicos (nombres y apellidos), perfil, usuario, clave de acceso y estado (activo / inactivo); si es inactivo no puede ingresar al sistema
5. Todas las entidades menos el detalle del pedido requieren un campo de observaciones
6. El perfil de administrador puede ver todas las funcionalidades y el perfil de vendedor todo menos la administración de usuarios.
7. El sistema debe escribir en un archivo plano el log del sistema
8. El sistema debe leer de un archivo plano el usuario y la clave de la base de datos

Entregables

|  |  |  |
| --- | --- | --- |
| **Actividad** | **Fecha** | **Peso** |
| **Segundo Corte** | **22-04** | **80%** |
| Modelo Relacional y Modelo de Clases | 25-03 | 20% |
| Diagrama de procesos / Diagrama de flujo y Mockups | 01-04 | 20% |
| Formulario de usuarios y Acceso de Administrador | 08-04 | 40% |
| Formulario de Clientes | 15-04 | 20% |
| **Tercer Corte** | **27-05** | **80%** |
| Formulario Productos | 29-04 | 20% |
| Formulario Pedidos | 13-05 | 30% |
| Proyecto completo (los dos perfiles) | 20-05 | 50% |

En cada una de las fechas estipuladas se hará la sustentación y revisión del avance, para la presentación y la retroalimentación debe estar el grupo completo

**GUIA PARA A ELABORACION DEL PROYECTO EN .NET #C**

|  |  |
| --- | --- |
|  | Se desarrollará el formulario para administrar Categorías, basado en la entidad categoría del modelo relacional  Como primera instancia en el motor de base de datos se deben crear los procedimientos almacenados que se utilizaran para la persistencia, en los métodos de Listar todas Categorías, Buscar Categoría por Id, Insertar, Actualizar y Eliminar. |

USE Tienda

GO

CREATE PROCEDURE ListarCategorias

AS

SELECT IdCategoria [Id],

    ISNULL(CodigoCategoria,'') [Codigo],

    ISNULL(Nombre,'')[Nombre],

    ISNULL(Observacion,'')[Observacion]

FROM Categoria

GO

CREATE PROCEDURE TraerCategoriaPorId

@ID int

AS

SELECT IdCategoria [Id],

    ISNULL(CodigoCategoria,'') [Codigo],

    ISNULL(Nombre,'')[Nombre],

    ISNULL(Observacion,'')[Observacion]

FROM Categoria

WHERE IdCategoria = @ID

GO

CREATE PROCEDURE InsertarCategoria

    @CODIGO VARCHAR(8),

    @NOMBRE VARCHAR(100),

    @OBSERVACION TEXT

AS

INSERT INTO Categoria(CodigoCategoria,Nombre,Observacion)

VALUES(@CODIGO,@NOMBRE,@OBSERVACION)

GO

CREATE PROCEDURE ActualizarCategoria

    @ID INT,

    @CODIGO VARCHAR(8),

    @NOMBRE VARCHAR(100),

    @OBSERVACION TEXT

AS

UPDATE Categoria SET CodigoCategoria = @CODIGO,

    Nombre = @NOMBRE, Observacion = @OBSERVACION

WHERE IdCategoria = @ID

GO

CREATE PROCEDURE EliminarCategoria

    @ID INT

AS

DELETE FROM Categoria

WHERE IdCategoria = @ID

GO

Ahora, vamos a construir el formulario de administración de categorías

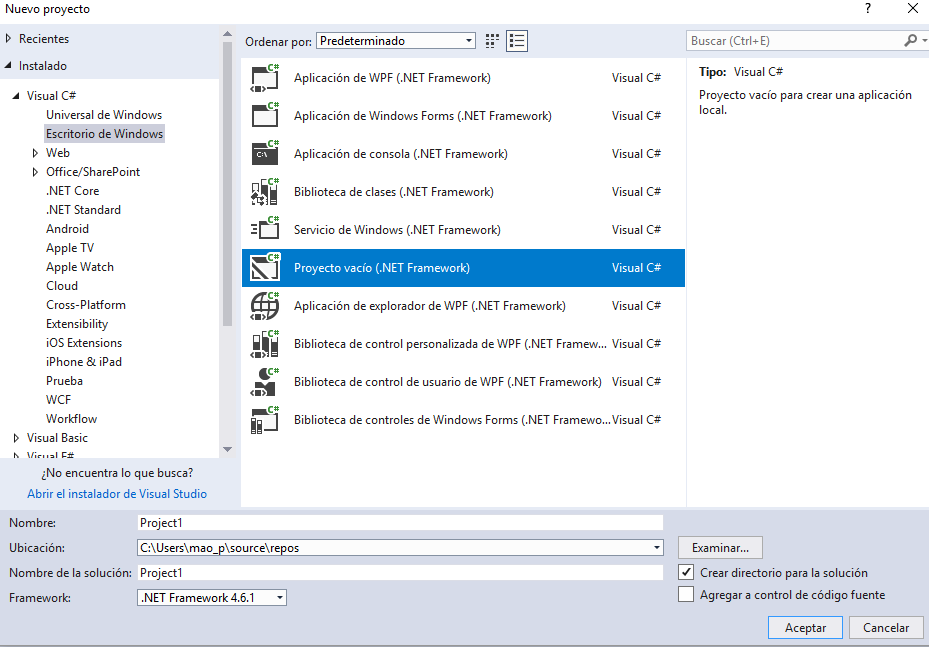
1. Crear una solución en blanco

Ingrese al menú Archivo => Nuevo => Proyecto

Visual C# => Escritorio de Windows =>Proyecto Vacío

Nombre de la solución Tienda.

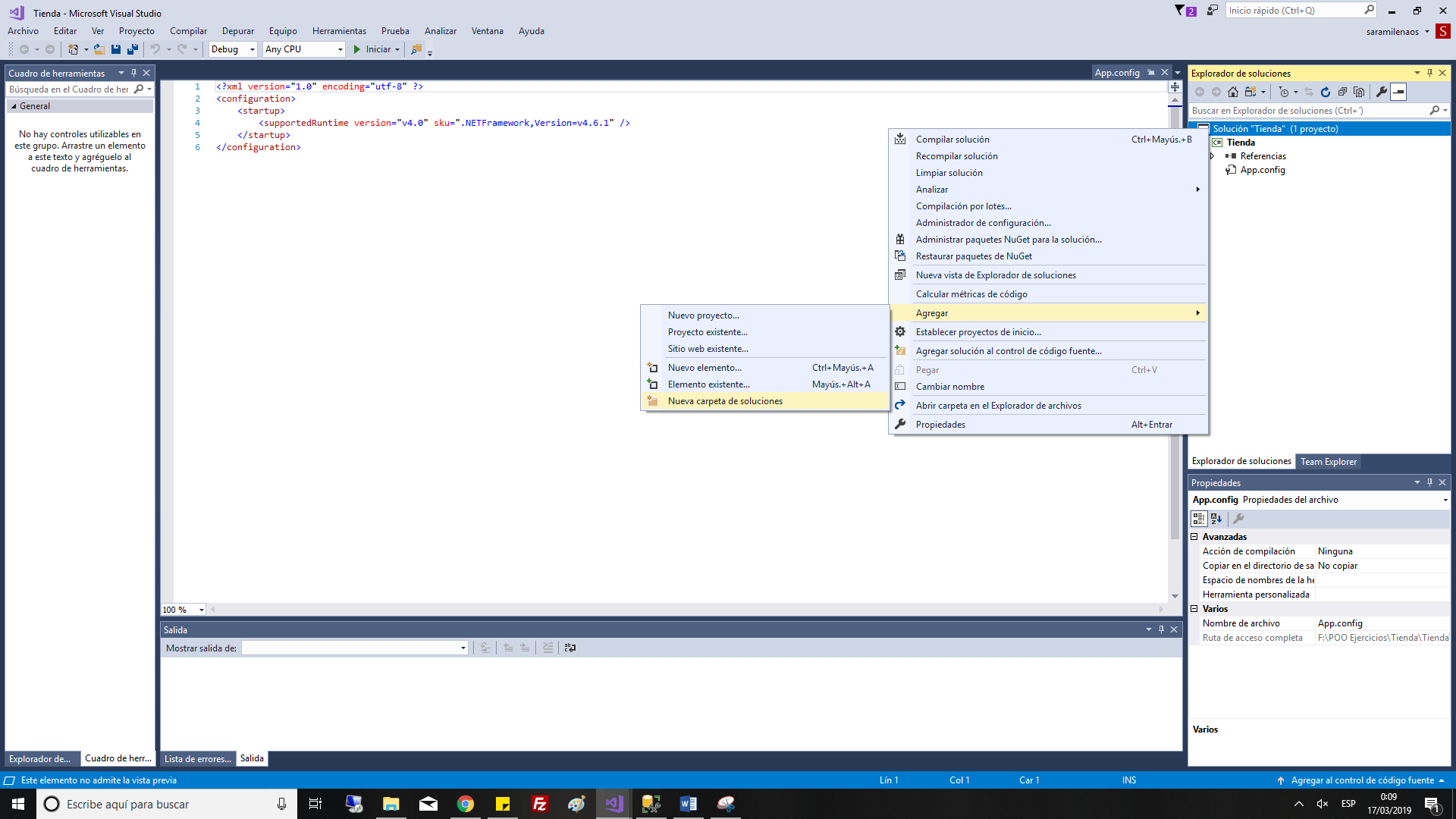
Seleccione la ubicación del proyecto



1. Agregar a la solución las carpetas Modelo y Aplicaciones.

Clic derecho sobre la solución => agregar => nueva carpeta de soluciones

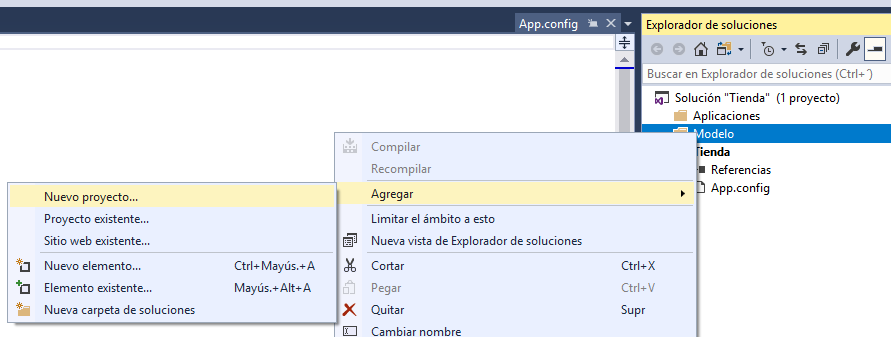
Asignar el nombre Modelo



|  |  |
| --- | --- |
| Clic derecho sobre la solución => agregar => nueva carpeta de soluciones  Asignar el nombre Aplicaciones |  |

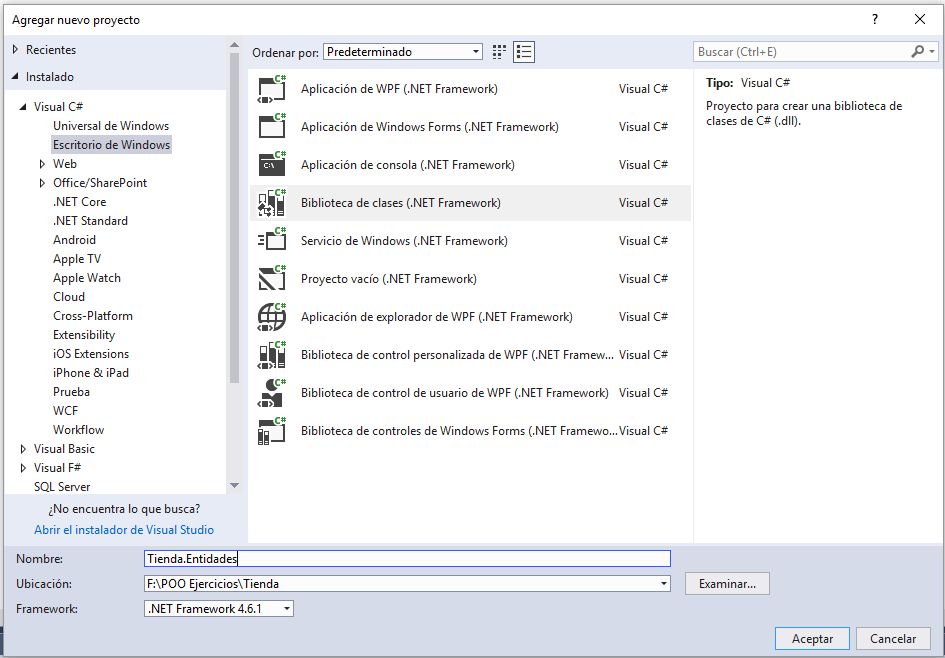
1. Crear la capa de entidad.

Sobre la Carpeta Modelo, clic derecho => agregar => nuevo proyecto

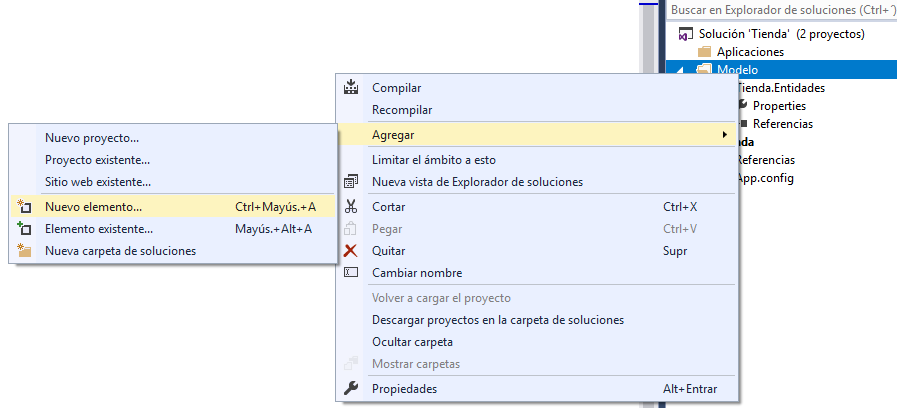


Seleccione escritorio de windows => biblioteca de clases como tipo de proyecto.

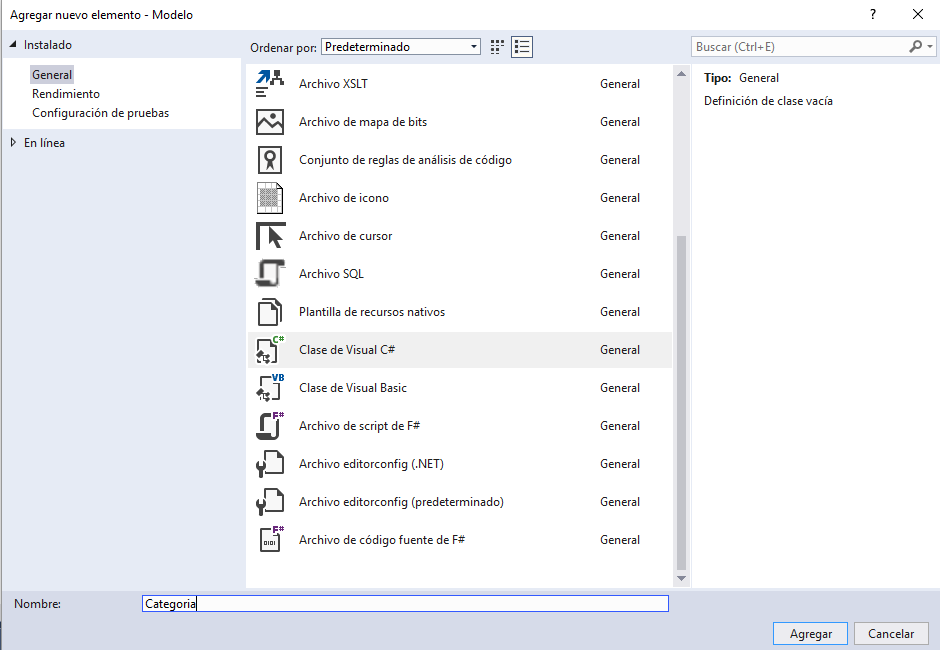
Asigne el nombre Tienda.Entidades



|  |  |
| --- | --- |
|  | Elimine la clase que se crea por defecto (Class1.cs)  Agregue un nuevo Item al proyecto, clic derecho => agregar => nuevo elemento |



Seleccione e tipo Clase de visual C# y asigne el nombre Categoria



Para que la clase sea visible para los demás proyectos le agregamos el modificador public

|  |  |
| --- | --- |
|  | public class Categoria |

Creamos los campos y métodos de la clase.

|  |  |
| --- | --- |
|  | // Atributos de la clase Categoria  int \_id;  string \_codigo;  string \_nombre;  string \_observacion;  // Se definen los metodos get y set  public int Id  {  get { return \_id; }  set { \_id = value; }  }    public string Codigo  {  get { return \_codigo; }  set { \_codigo = value; }  }  public string Nombre  {  get { return \_nombre; }  set { \_nombre = value; }  }  public string Observacion  {  get { return \_observacion; }  set { \_observacion = value; }  } |
|  |  |

Creamos los constructores de la clase.

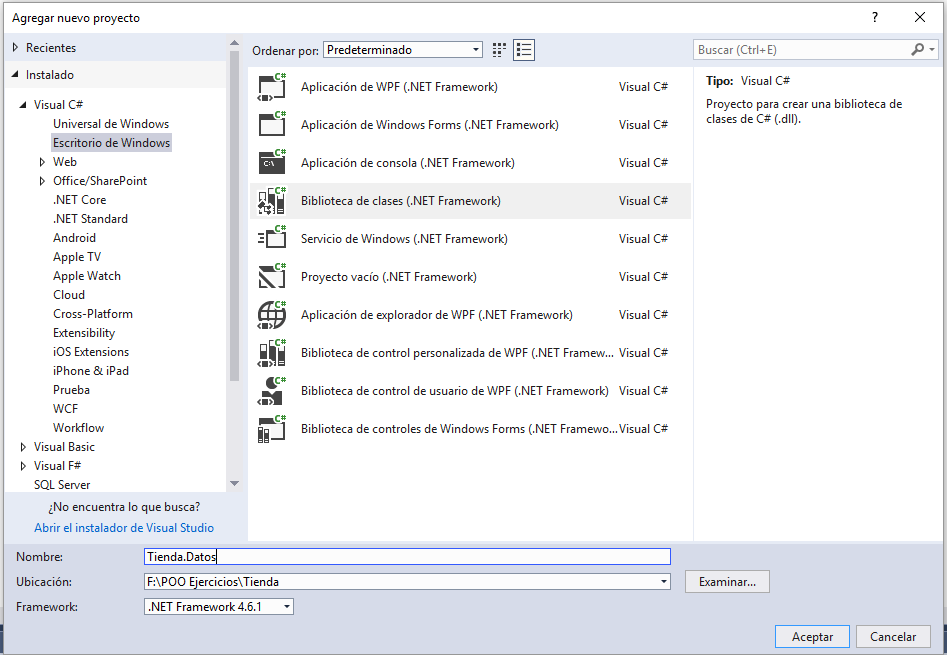
|  |  |
| --- | --- |
|  | public Categoria(int Id, string Codigo, string Nombre, string Observacion)  {  this.\_id = Id;  this.\_codigo = Codigo;  this.\_nombre = Nombre;  this.\_observacion = Observacion;  }  public Categoria(int Id, string Codigo, string Nombre) : this(Id, Codigo, Nombre, "")  {  }  public Categoria(int Id, string Nombre): this(Id, "", Nombre, "")  {  }  public Categoria(): this(0, "", "", "")  {  } |

1. Crear la capa de Datos.

Sobre la Carpeta Modelo, clic derecho => agregar => nuevo proyecto

Seleccione Escritorio de Windows => biblioteca de clases como tipo de proyecto.

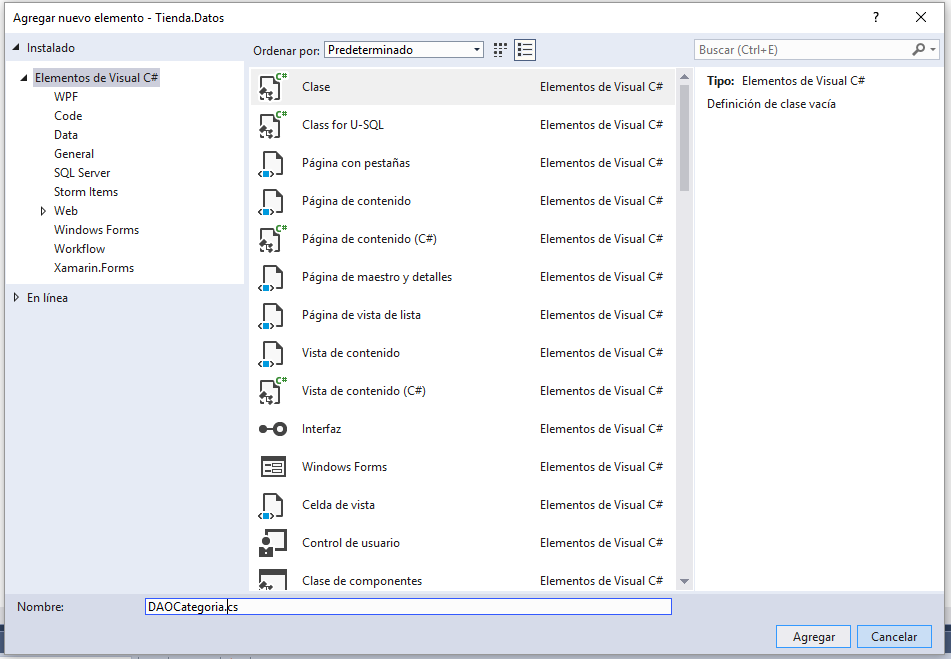
Asigne el nombre Tienda.Datos



Elimine la clase que se crea por defecto (Class1.cs)

Agregue un nuevo Item al proyecto, clic derecho => agregar => nuevo elemento

Seleccione el tipo Class y asigne el nombre DAOCategoria

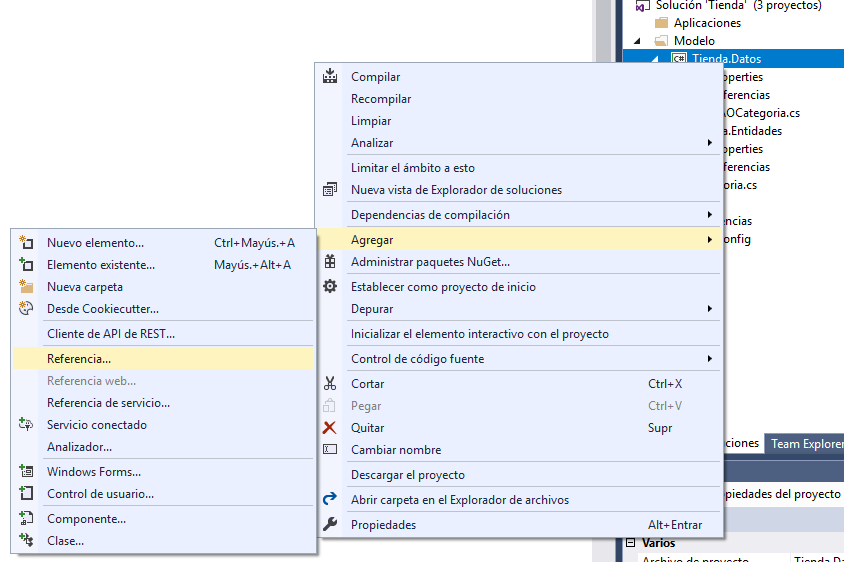


Para que la clase sea visible para los demás proyectos le agregamos el modificador public

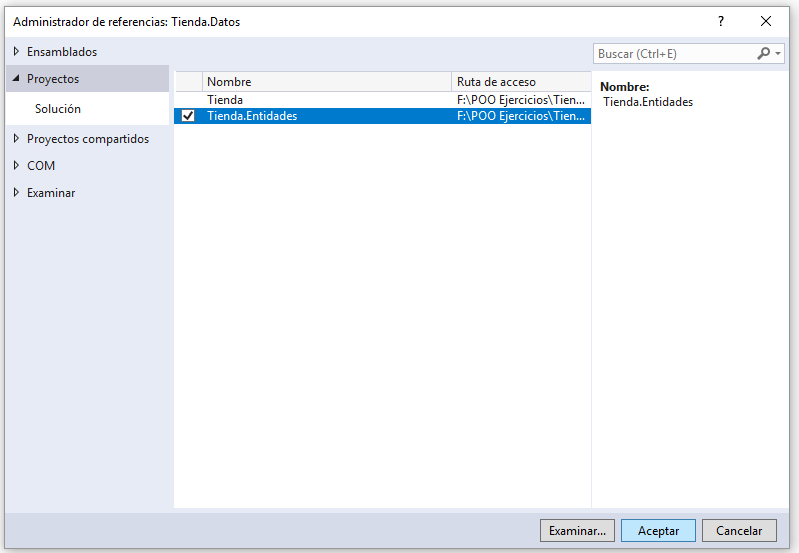
|  |  |
| --- | --- |
|  | public class DAOCategoria |

Agregue la referencia al proyecto:

Proyecto Tienda.Datos, clic derecho => Agregar => Referencia,



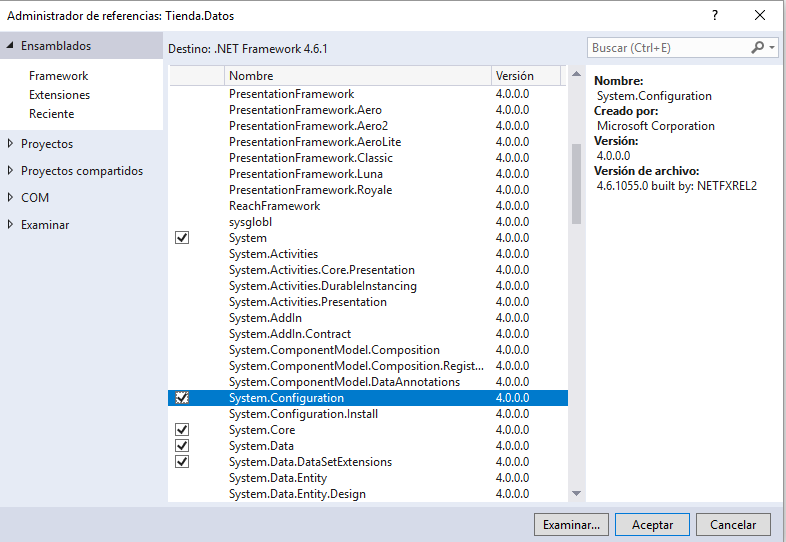
Seleccione la ficha Proyectos, luego ubique el proyecto Tienda.Entidades => Aceptar



|  |  |
| --- | --- |
| Importamos la referencia para esto agregamos la siguiente instrucción.  using Tienda.Entidades; |  |

|  |  |
| --- | --- |
|  |  |

Vamos a crear una propiedad que tome la cadena de conexión a la base de datos, esta cadena de conexión será definida más adelante en el archivo de configuración de la aplicación sea web (webconfig) o Windows (AppConfig), para tomar esta configuración hacemos sobre el Proyecto Tienda.Datos, clic derecho => agregar referencia, Seleccione la ficha Ensamblados, luego ubique System.Configuration y Aceptar. (esto solo se hace una vez)



Ahora sobre la clase DAOCategoria, importamos la referencia System.Configuration, para la conexión a la base de datos y System.Data, para acceder a la librería de Datos y a SQLServer.

|  |  |
| --- | --- |
|  | using System.Configuration;  using System.Data.SqlClient;  using System.Data; |

Creamos una propiedad llamada CadenaConexion, para este caso preguntamos si es que el campo \_cadenaConexion recuperó el valor de un key llamado Conex que estará definido en el archivo de configuración.

|  |  |
| --- | --- |
|  | string \_cadenaConexion;  public string CadenaConexion  {  get  {  if (\_cadenaConexion == null)  {  \_cadenaConexion = ConfigurationManager.  ConnectionStrings["Conex"].ConnectionString;  }  return \_cadenaConexion;  }  set { \_cadenaConexion = value; }  } |

Ahora creamos los métodos de acceso a datos, tales como Listar, Insertar, Eliminar, Actualizar.

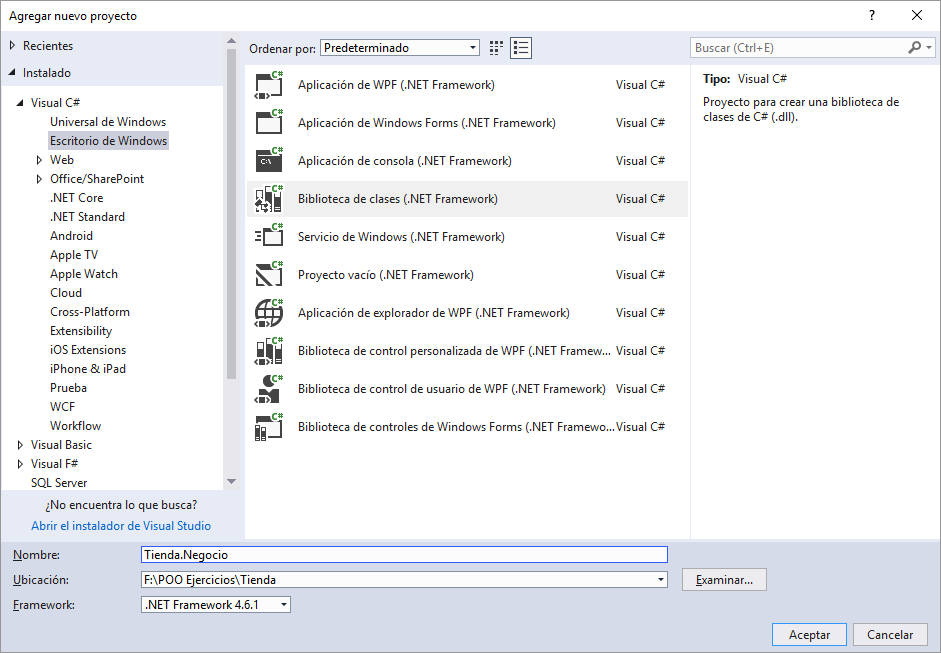
|  |  |
| --- | --- |
|  | public List<Categoria> Listar()  {  List<Categoria> lista = new List<Categoria>();  using (SqlConnection con = new SqlConnection(CadenaConexion))  {  con.Open();  SqlCommand cmd = new SqlCommand("ListarCategorias", con);  cmd.CommandType = CommandType.StoredProcedure;  SqlDataReader dr = cmd.ExecuteReader();  if (dr != null && dr.HasRows)  {  while (dr.Read())  {  Categoria c = new Categoria((int)dr["Id"],  (string)dr["Codigo"], (string)dr["Nombre"],  (string)dr["Observacion"]);  lista.Add(c);  }  }  }  return lista;  }  public Categoria TraerPorId(int Id)  {  Categoria Categoria = new Categoria();  using (SqlConnection con = new SqlConnection(CadenaConexion))  {  con.Open();  SqlCommand cmd = new SqlCommand("TraerCategoriaPorId", con);  cmd.CommandType = CommandType.StoredProcedure;  cmd.Parameters.AddWithValue("@ID", Id);  SqlDataReader dr = cmd.ExecuteReader();  if (dr != null && dr.HasRows)  {  dr.Read();  Categoria = new Categoria((int)dr["Id"],  (string)dr["Codigo"], (string)dr["Nombre"],  (string)dr["Observacion"]);  }  }  return Categoria;  }  public int Insertar(Categoria Categoria)  {  int n = -1;  using (SqlConnection con = new SqlConnection(CadenaConexion))  {  con.Open();  SqlCommand cmd = new SqlCommand("InsertarCategoria", con);  cmd.CommandType = CommandType.StoredProcedure;  cmd.Parameters.AddWithValue("@Codigo", Categoria.Codigo);  cmd.Parameters.AddWithValue("@Nombre", Categoria.Nombre);  cmd.Parameters.AddWithValue("@Observacion", Categoria.Observacion);  n = cmd.ExecuteNonQuery();  }  return n;  }  public int Actualizar(Categoria Categoria)  {  int n = -1;  using (SqlConnection con = new SqlConnection(CadenaConexion))  {  con.Open();  SqlCommand cmd = new SqlCommand("ActualizarCategoria", con);  cmd.CommandType = CommandType.StoredProcedure;  cmd.Parameters.AddWithValue("@Id", Categoria.Id);  cmd.Parameters.AddWithValue("@Codigo", Categoria.Codigo);  cmd.Parameters.AddWithValue("@Nombre", Categoria.Nombre);  cmd.Parameters.AddWithValue("@Observacion", Categoria.Observacion);  n = cmd.ExecuteNonQuery();  }  return n;  }  public int Eliminar(int Id)  {  int n = -1;  using (SqlConnection con = new SqlConnection(CadenaConexion))  {  con.Open();  SqlCommand cmd = new SqlCommand("EliminarCategoria", con);  cmd.CommandType = CommandType.StoredProcedure;  cmd.Parameters.AddWithValue("@Id", Id);  n = cmd.ExecuteNonQuery();  }  return n;  } |

1. Crear la capa de Negocio.

Sobre la Carpeta Modelo, clic derecho => agregar => nuevo proyecto

Seleccione escritorio de Windows => biblioteca de clases como tipo de proyecto.

Asigne el nombre Tienda.Negocio



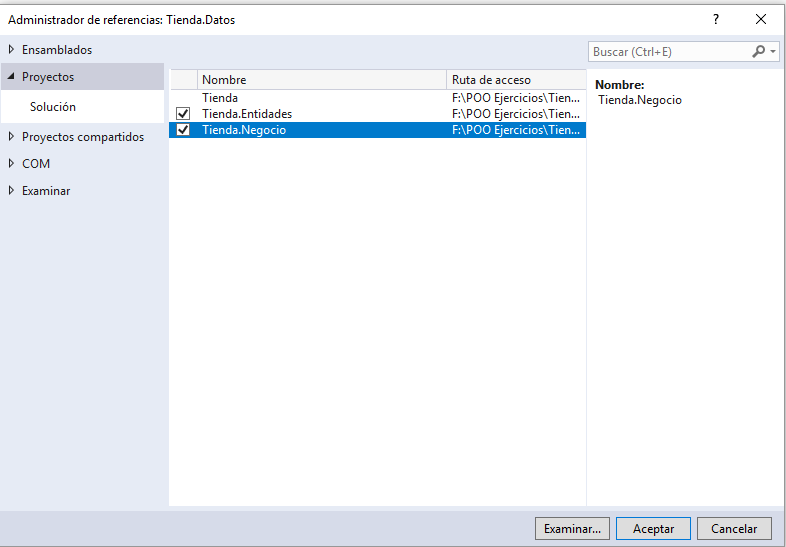
|  |  |
| --- | --- |
|  | Elimine la clase que se crea por defecto (Class1.cs)  Agregue un nuevo Item al proyecto, clic derecho => agregar => nuevo elemento  Seleccione el tipo Class y asigne el nombre BLCategoria |

Para que la clase sea visible para los demás proyectos le agregamos el modificador public

|  |  |
| --- | --- |
|  | public class BLCategoria |

@) Agregue la referencia al proyecto:

Tienda.Datos, clic derecho => Agregar referencia, Seleccione la ficha Proyectos, luego seleccione el proyecto Tienda.Entidades y Tienda.Datos => Aceptar



Importamos la referencia para esto agregamos la siguiente instrucción.

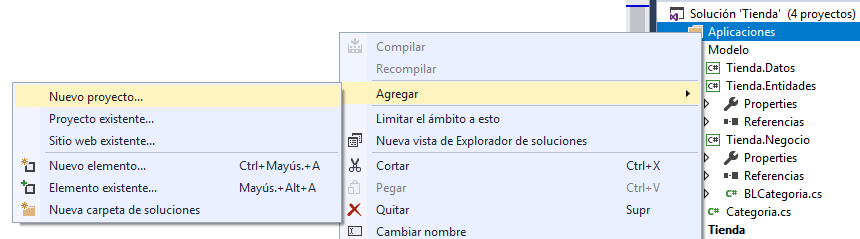
|  |  |
| --- | --- |
|  | using Tienda.Entidades;  using Tienda.Datos; |

Creamos ahora nuestra lógica de negocio, para esto creamos los métodos tales como Listar, TraerPorId, Insertar, Eliminar, Actualizar.

|  |  |
| --- | --- |
|  | public List<Categoria> Listar()  {  DAOCategoria daCategoria = new DAOCategoria();  return daCategoria.Listar();  }    public Categoria TraerPorId(int Id)  {  DAOCategoria daCategoria = new DAOCategoria();  return daCategoria.TraerPorId(Id);  }    public int Insertar(Categoria Categoria)  {  DAOCategoria daCategoria = new DAOCategoria();  return daCategoria.Insertar(Categoria);  }    public int Actualizar(Categoria Categoria)  {  DAOCategoria daCategoria = new DAOCategoria();  return daCategoria.Actualizar(Categoria);  }    public int Eliminar(int Id)  {  DAOCategoria daCategoria = new DAOCategoria();  return daCategoria.Eliminar(Id);  } |

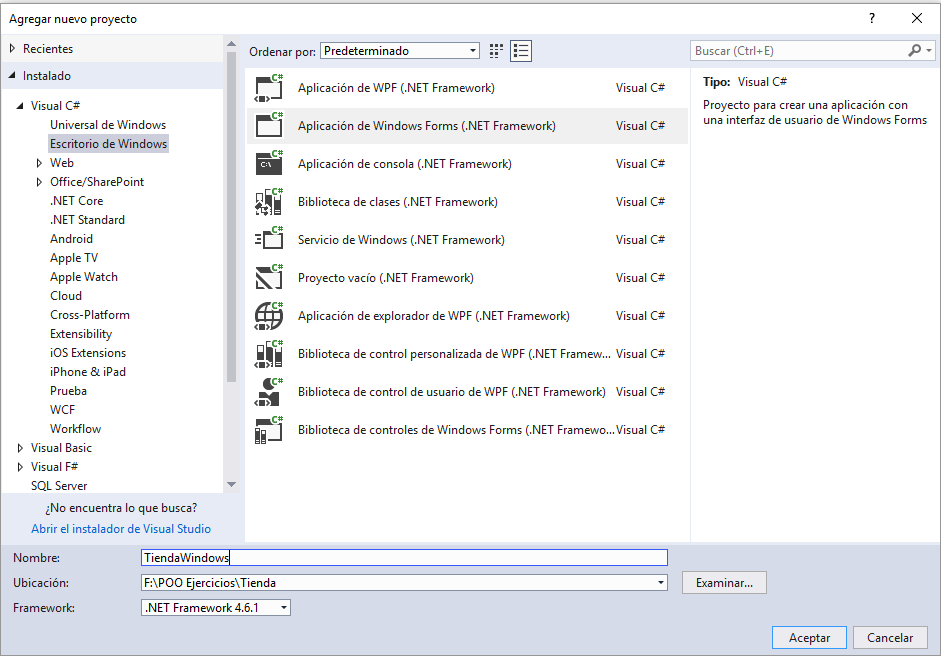
1. Crear la Aplicación.

Sobre la Carpeta Aplicaciones, clic derecho => agregar => nuevo proyecto

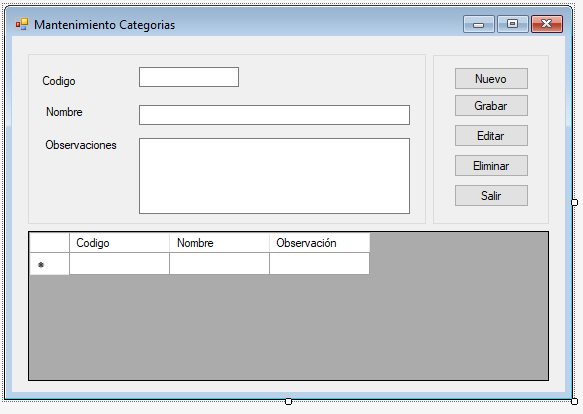


Seleccione escritorio de windows => aplicación de Windows Forms como tipo de proyecto.

Asigne el nombre TiendaWindows



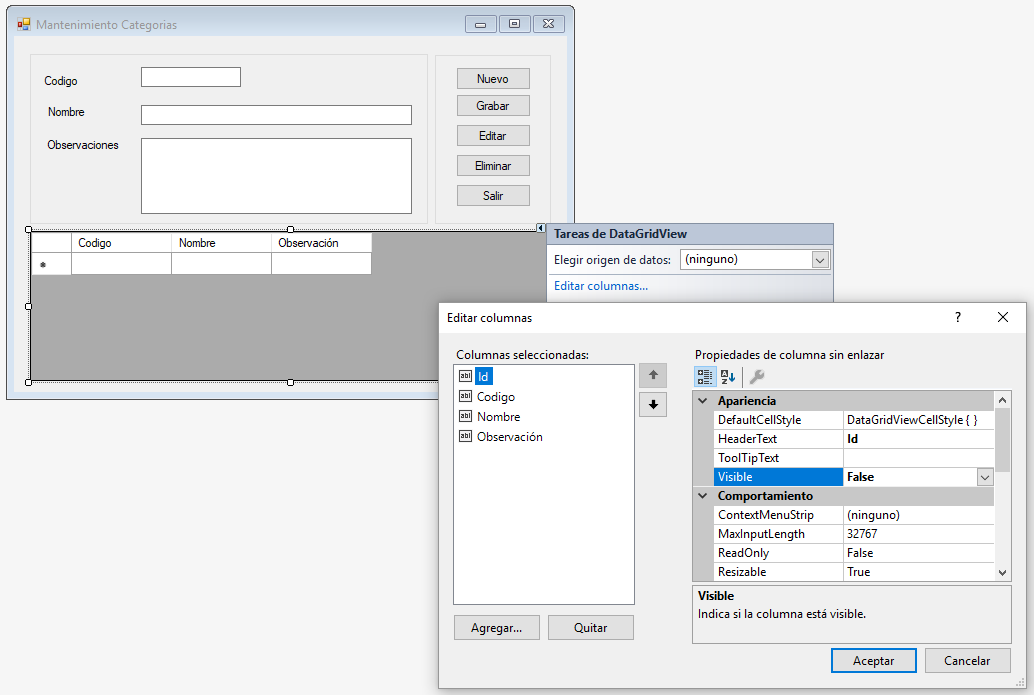
Diseñe el siguiente formulario

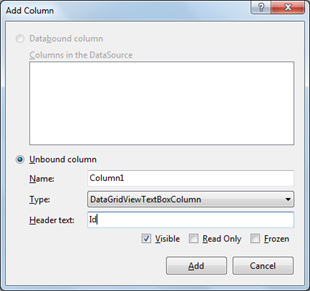


En el GroupBox gbDatos: Contiene a todos los label y TextBox (txtCodigo; txtNombre; txtObservacion); el txtObservacion debería mostrarse multilinea

En el GroupBox gbControl: contiene a los Button (btnNuevo, btnGrabar, btnEditar, btnEliminar, btnSalir)

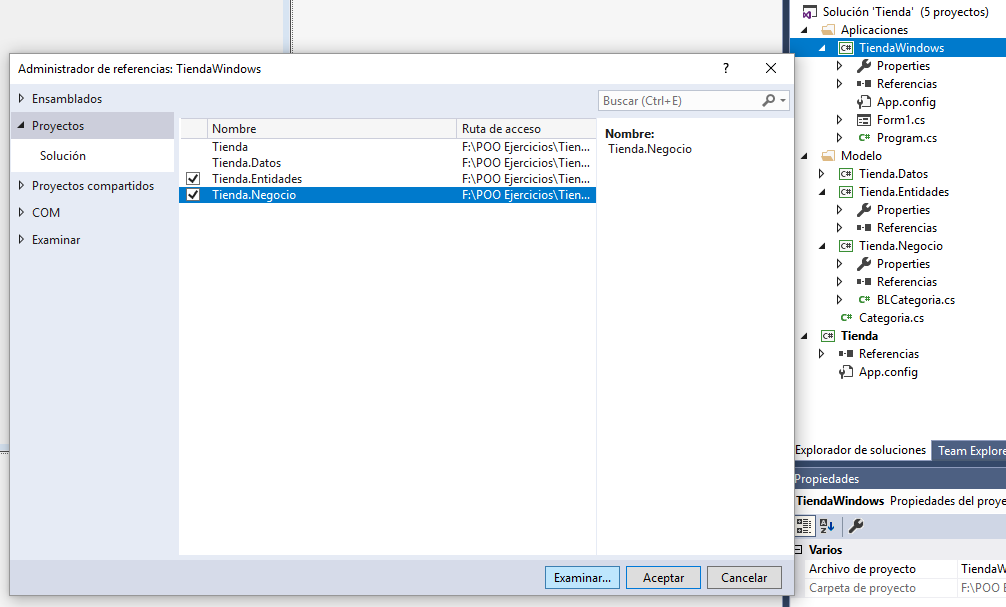
**DataGridView** dgvDatos; Personalize el control dgvDatos, agregue las columnas Id, Código, Nombre, Observación. Modifique la propiedad Visible=False de la columna Id.



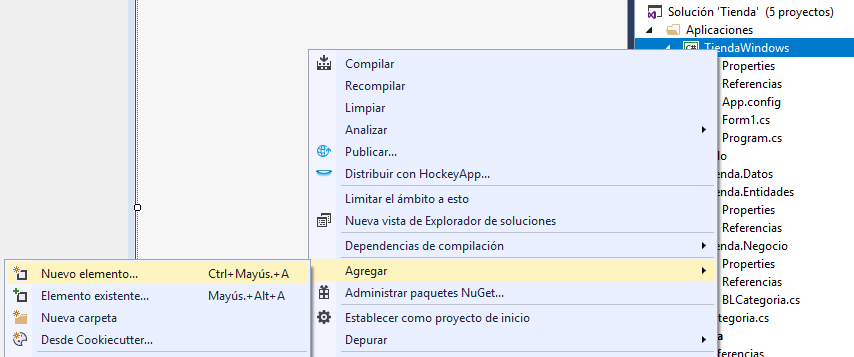


@) Agregue la referencia al proyecto:

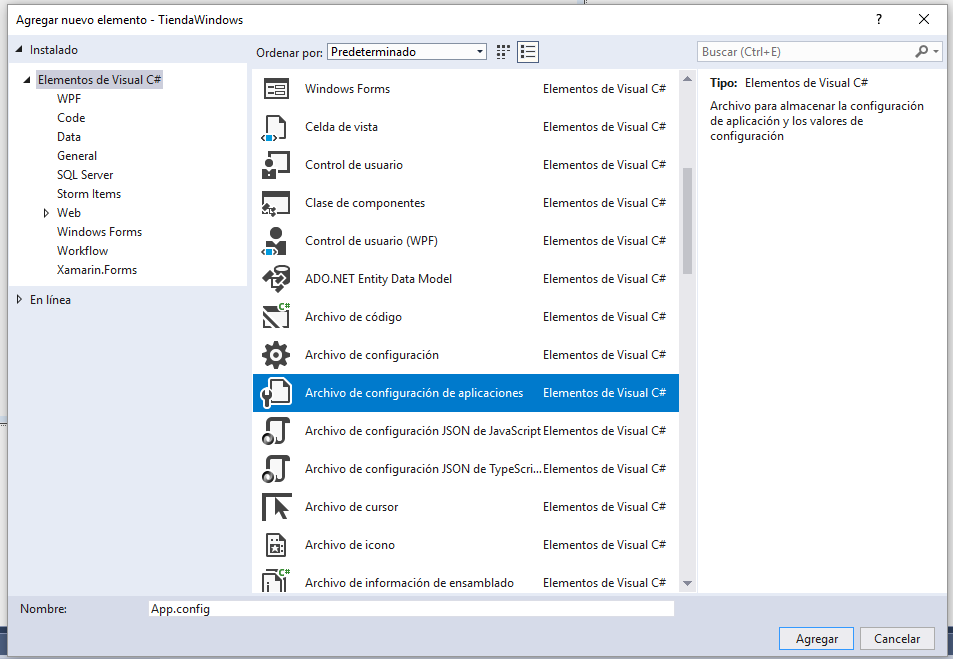
Proyecto TiendaWindows, clic derecho => Agregar =>Referencia, Seleccione la ficha Proyectos, luego seleccione el proyecto Tienda.Entidades y Tienda.Negocio => Aceptar



Agregue un nuevo Item al proyecto, clic derecho => Agregar => nuevo elemento



Seleccione e tipo Application Configuration File, el nombre será por defecto App.config.



Abrimos el archivo App.config y agregamos la siguiente instrucción después de <configuration>.

|  |  |
| --- | --- |
|  | <connectionStrings>  <add name="Conex" connectionString="Data Source=NOMBRE\_SERVIDOR;  Initial Catalog=Tienda; UID = usuarioBD; PWD = ClaveBD"/>  </connectionStrings> |

Importamos la referencia para esto agregamos la siguiente instrucción.

|  |  |
| --- | --- |
|  | using Tienda.Entidades;  using Tienda.Negocio; |

Declaramos las variables generales

|  |  |
| --- | --- |
|  | List<Categoria> lista = null;  BLCategoria blCategoria = new BLCategoria();  Categoria c;  bool \_nuevo = false;  public Cartegorias()  {  InitializeComponent();  ActivarControlDatos(gbDatos, false);  CargarDatos();  } |

Creamos un método que permita activar los controles

|  |  |
| --- | --- |
|  | private void ActivarControlDatos(Control Contenedor, bool Estado)  {  try  {  foreach (var item in Contenedor.Controls)  {  if (item.GetType() == typeof(TextBox))  {  ((TextBox)item).Enabled = Estado;  }  }  }  catch (Exception)  {  }  } |

Creamos un método que limpie los controles

|  |  |
| --- | --- |
|  | private void LimpiarControl(Control Contenedor)  {  foreach (var item in Contenedor.Controls)  {  if (item.GetType() == typeof(TextBox))  {  ((TextBox)item).Clear();  }  }  } |

Creamos un método que Active los Botones

|  |  |
| --- | --- |
|  | private void ActivarButton(bool Estado)  {  btnNuevo.Enabled = Estado;  btnGrabar.Enabled = !Estado;  btnEliminar.Enabled = Estado;  btnSalir.Enabled = Estado;  } |

Creamos el método CargarDatos

|  |  |
| --- | --- |
|  | private void CargarDatos()  {  if (lista == null)  {  lista = blCategoria.Listar();  }  if (lista.Count > 0)  {  dgvDatos.Rows.Clear();  for (int i = 0; i < lista.Count; i++)  {  dgvDatos.Rows.Add(lista[i].Id, lista[i].Codigo,  lista[i].Nombre, lista[i].Observacion);  }  }  } |

Botón Nuevo

|  |  |
| --- | --- |
|  | private void btnNuevo\_Click(object sender, EventArgs e)  {  \_nuevo = true;  ActivarControlDatos(gbDatos, true);  btnEditar.Text = "Cancelar";  ActivarButton(false);  LimpiarControl(gbDatos);  txtCodigo.Focus();  } |

Botón Grabar

|  |  |
| --- | --- |
|  | private void btnGrabar\_Click(object sender, EventArgs e)  {  int n = -1;  if (\_nuevo)  {  c = new Categoria(0, txtCodigo.Text,  txtNombre.Text, txtObservacion.Text);  n = blCategoria.Insertar(c);  }  else  {  c.Codigo = txtCodigo.Text;  c.Nombre = txtNombre.Text;  c.Observacion = txtObservacion.Text;  n = blCategoria.Actualizar(c);  }  if (n > 0)  {  MessageBox.Show("Datos grabados correctamente", "Aviso",  MessageBoxButtons.OK, MessageBoxIcon.Information);  ActivarControlDatos(gbDatos, false);  ActivarButton(true);  dgvDatos.Enabled = true;  LimpiarControl(gbDatos);  btnEditar.Text = "Editar";  lista = blCategoria.Listar();  CargarDatos();  }  else  {  MessageBox.Show("Error al grabar", "Aviso",  MessageBoxButtons.OK, MessageBoxIcon.Error);  }  } |

Botón Editar

|  |  |
| --- | --- |
|  | private void btnEditar\_Click(object sender, EventArgs e)  {  \_nuevo = false;  if (btnEditar.Text == "Cancelar")  {  LimpiarControl(gbDatos);  ActivarControlDatos(gbDatos, false);  ActivarButton(true);  dgvDatos.Enabled = true;  btnEditar.Text = "Editar";  }  else  {  if (dgvDatos.RowCount > 0)  {  c = blCategoria.TraerPorId((int)dgvDatos[0, dgvDatos.  CurrentRow.Index].Value);  txtCodigo.Text = c.Codigo;  txtNombre.Text = c.Nombre;  txtObservacion.Text = c.Observacion;  ActivarControlDatos(gbDatos, true);  ActivarButton(false);  dgvDatos.Enabled = false;  btnEditar.Text = "Cancelar";  }  }  } |

Botón Eliminar

|  |  |
| --- | --- |
|  | private void btnEliminar\_Click(object sender, EventArgs e)  {  if (dgvDatos.RowCount > 0)  {  c = blCategoria.TraerPorId((int)dgvDatos[0, dgvDatos.  CurrentRow.Index].Value);  DialogResult rpta =  MessageBox.Show("Desea eliminar el registro", "Eliminar",  MessageBoxButtons.YesNo, MessageBoxIcon.Question);  if (rpta == System.Windows.Forms.DialogResult.Yes)  {  int n = blCategoria.Eliminar(c.Id);  if (n > 0)  {  MessageBox.Show("Registro eliminado", "Aviso",  MessageBoxButtons.OK, MessageBoxIcon.Information);  lista = blCategoria.Listar();  CargarDatos();  }  else  {  MessageBox.Show("Error al eliminar", "Aviso",  MessageBoxButtons.OK, MessageBoxIcon.Error);  }  }  }  } |

Botón Salir

|  |  |
| --- | --- |
|  | private void btnSalir\_Click(object sender, EventArgs e)  {  Close();  } |

Ejecutar la Aplicación

