**E4PROYECTO JARDIN INFANTIL “*MIS PRIMERAS LETRAS”***

**LENGUAJE DE PROGRAMACION (ASSOCIATE)**

Se debe crear un sistema de información de escritorio para el Jardín Infantil “Mis primeras letras”; donde cumpla con las siguientes especificaciones técnicas y funcionales:

**Especificaciones Técnicas y/o no Funcionales**

1. Se debe construir una herramienta Windows, ambiente cliente/servidor.
2. Se debe implementar en lenguaje de programación .NET C#.
3. El motor de base de datos debe ser SQLServer Express. La Base debe ser centralizada.
4. La arquitectura debe ser en Capas (MVC).
5. El sistema debe contar con un módulo de seguridad, que permita la administración de los usuarios y acceso por perfiles.
6. El sistema debe contar con la definición y aplicación de los atributos de calidad de Software.
7. El sistema debe contar con registro de auditoria.

**Entregables**

1. Se debe elaborar manual de usuario.
2. Se debe elaborar el documento técnico; debe incluir el modelo 4+1
3. Se debe elaborar el modelo relacional de la base de datos y la documentación incluyendo metadatos.

**Requerimientos Funcionales**

El sistema debe contar con las siguientes condiciones y características basadas en la operación del Jardín Infantil:

1. El sistema debe contar con tres perfiles
   * **Administrador**: debe poder administrar los usuarios, los grados, los grupos, áreas, materias, salones y periodos; así como generar los reportes de notas, pagos y listados de estudiantes por grupos.
   * **Secretaria**: debe poder administrar los docentes, asignar docentes por materia en cada grupo, inscribir los estudiantes a cada grupo y registrar los pagos de matrícula, pensión y otros; así como generar los reportes de pagos y listados de estudiantes por grupo.
   * **Docente**: debe poder crear los logros de su materia e ingresar las notas de los estudiantes; así como generar los reportes de notas y listados de estudiantes por materia.
2. Los módulos del sistema deben ser:

* **Modulo administrador de usuarios**: **Rol Administrador.** Debe permitir crear, modificar y eliminar usuarios; asignarles el rol que tendrá en el sistema. Así, como permitirle el ingreso al sistema, el bloqueo y activación, cambio de contraseña.
* **Módulo de gestión**: **Rol Administrador.** Debe permitir la creación de Grados (como por ejemplo transición, kínder), Grupos (a cada grado se le deben asociar los grupos escolares; por ejemplo, kínder A y kínder B; por grado son máximo dos grupos), Áreas y Materias (las materias deben pertenecer a un área)

o    CIENCIAS NATURALES (Biología 2)

o    CIENCIAS SOCIALES (Sociales 2)

o    EDUCACIÓN ARTÍSTICA (Artes Plásticas 2, Música 1)

o    EDUCACIÓN ÉTICA (Ética 2)

o    EDUCACIÓN RELIGIOSA (Religión 1)

o    EDUCACIÓN FÍSICA, RECREACIÓN Y DEPORTE (Educación Física 2, Danzas 1)

o    HUMANIDADES (Lengua Castellana 3, inglés 2)

o    MATEMÁTICAS (Matemáticas 4)

o    TECNOLOGÍA E INFORMÁTICA (Informática 2)

* **Módulo de matrículas**: **Rol Secretaria.** Debe permitir a la secretaria realizar el registro de estudiantes en cada grupo. Por grupo máximo son 20 estudiantes.
* **Módulo de pagos**: **Rol Secretaria.** Debe permitir a la secretaria realizar el registro de los pagos realizados por los estudiantes; este deberá registrar el abono y el saldo vrs el valor total a pagar. Debe existir un diccionario con cada uno de los pagos a realizar, donde este el valor total a pagar.
* **Asignación de carga académica**: **Rol Administrador.** Debe permitir la asignación de Grupo, Materia, Salón, día, hora y Docente; con el fin de distribuir la carga académica.

Se debe tener en cuenta las siguientes validaciones:

* + No se puede asignar más de una vez un salón en el mismo día y hora.
  + Un docente no puede estar en dos grupos en el mismo momento.
  + A un grupo máximo se le podrán asignar 5 horas diarias.
  + Las horas académicas son de 7 am a 3 pm.
  + Una materia se podrá asignar máximo el número horas determinadas a la semana, como en el ejemplo:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Biología** | **Sociales** | **Artes** | **Música** | **Ética** | **Religión** | **Edu Física** | **Danzas** | **Lengua Castellana** | **Ingles** | **Matemáticas** | **Informática** |
| **7 a 8** | **M** | **V** |  |  | **L** |  |  |  | **J** |  | **W** |  |
| **8 a 9** | **M** | **V** |  |  | **L** |  |  |  | **J** |  | **W** |  |
| **9 a 10** | **DESCANSO** | | | | | | | | | | | |
| **11 a 12** |  |  | **M** |  |  |  | **V** |  |  | **W** | **L** | **J** |
| **12 a 1** |  |  | **M** |  |  |  | **V** |  |  | **W** | **L** | **J** |
| **1 a 2** | **DESCANSO** | | | | | | | | | | | |
| **2 a 3** |  |  |  | **L** |  | **V** |  | **J** | **W** | **M** |  |  |

* + Se debe generar el reporte del horario por grupo
* **Módulo de Notas**: **Rol Docente**. Debe permitir ingresar las notas de cada periodo académico por materia, teniendo en cuenta que se deberá ingresar una nota de 0 a 10, donde se valoraran según la escala correspondiente para el informe académico. Es decir, se ingresará el valor, pero se debe mostrar en el informe la valoración.

|  |  |  |
| --- | --- | --- |
| **VALORACION** |  | **VALOR** |
| Insuficiente | **<** | 6 |
| Aceptable | **<** | 8 |
| Sobresaliente | **<** | 9 |
| Excelente | **< =** | 10 |

Los valores serán tenidos en cuenta para el cálculo de la definitiva por periodo, pero no se deberán mostrar en el informe del periodo.

|  |  |  |
| --- | --- | --- |
| **MATERIA** | **VALOR** | **VALORACION** |
| **Biología** | 5,0 | Insuficiente |
| **Sociales** | 7,0 | Aceptable |
| **Artes** | 8,0 | Sobresaliente |
| **Música** | 7,5 | Aceptable |
| **Ética** | 8,9 | Sobresaliente |
| **Religión** | 9,5 | Excelente |
| **Edu. Física** | 9,0 | Excelente |
| **Danzas** | 10,0 | Excelente |
| **Lengua Castellana** | 8,0 | Sobresaliente |
| **Ingles** | 8,0 | Sobresaliente |
| **Matemáticas** | 6,0 | Aceptable |
| **Informática** | 10,0 | Excelente |
| **PROMEDIO** | **8,1** | **Sobresaliente** |

En cada periodo, se debe calcular el promedio total, para esto se debe tener en cuenta que cada uno de estos tiene un peso. Este informe solo se debe presentar al final del año como certificado del curso.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **30%** | **30%** | **40%** | **DEFINITIVA** | **VALORACION** |
| **Biología** | 5,0 | 6,0 | 8,0 | **6,5** | Aceptable |
| **Sociales** | 7,0 | 5,0 | 7,0 | **6,4** | Aceptable |
| **Artes** | 8,0 | 7,0 | 8,0 | **7,7** | Aceptable |
| **Música** | 7,5 | 8,0 | 7,5 | **7,7** | Aceptable |
| **Ética** | 8,9 | 7,5 | 8,9 | **8,5** | Sobresaliente |
| **Religión** | 9,5 | 8,9 | 9,5 | **9,3** | Excelente |
| **Edu. Física** | 9,0 | 9,5 | 9,0 | **9,2** | Excelente |
| **Danzas** | 10,0 | 9,0 | 8,0 | **8,9** | Sobresaliente |
| **Lengua Castellana** | 8,0 | 8,0 | 8,0 | **8,0** | Sobresaliente |
| **Ingles** | 8,0 | 8,0 | 8,0 | **8,0** | Sobresaliente |
| **Matemáticas** | 6,0 | 8,0 | 6,0 | **6,6** | Aceptable |
| **Informática** | 10,0 | 8,0 | 9,0 | **9,0** | Excelente |
| **PROMEDIO** | **8,1** | **7,7** | **8,1** | **8,0** | **Aceptable** |

Se debe almacenar el promedio general de cada estudiante, ya que con este se calcula el puesto que el estudiante ocupa en el salón ordenándolo de forma ascendente. Este puesto debe mostrarse en el boletín.

* **Módulo de reportes**: Se deben generar los siguientes reportes:
  + Reportes de notas por periodo (PDF)
  + Reportes de notas final (PDF)
  + Reporte de pagos por estudiante (PDF)
  + Listados de estudiantes por grupo (Excel)
  + Listados de estudiantes por materia (Excel)
  + Horario por grupo (Excel)
  + Reporte en XML del esquema de grado, grupo, áreas, materias
  + Listados de auditoria en archivo plano

1. Datos mínimos a capturar:

* **Alumnos:** apellido, nombre, fecha de nacimiento, acudiente, dirección, teléfono, correo, ocupación, observaciones
* **Materias:** materia, área, intensidad horaria
* **Docente:** apellido, nombre, dirección, teléfono, correo, observaciones

1. Todas las tablas deben contener campos de auditoria como usuario, fecha/hora de registro. La auditoría debe registrarse en archivos planos, donde se especifique cuando, quien, en que modulo y que acción se realizó, incluir el login.

**Entregables**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Actividad** | **Fecha** | | **Peso** | |
| **Primer Corte** | **Grupo 600** | **Grupo 800** | **30%** | |
| ***Parcial*** | ***10 marzo*** | ***11 marzo*** | | ***60%*** | |
| ***Proyecto*** | |  | | ***40%*** | |
| Modelo Relacional y Modelo de Clases | 18 febrero | 19 febrero | | 20% | |
| Diagrama de Flujo y Mockups | 25 febrero | 26 febrero | | 20% | |
| Entrega 1 – Admón. Usuarios | 3 marzo | 4 marzo | | 10% | |
| Entrega 2 – Gestión | 10 marzo | 11 marzo | | 50% | |
| **Segundo Corte** | **Grupo 600** | **Grupo 800** | **30%** | |
| ***Parcial*** | ***21 abril*** | ***22 abril*** | | ***20%*** | |
| ***Proyecto*** | |  | | ***80%*** | |
| Entrega 3 – Matriculas | 24 marzo | 25 marzo | | 25% | |
| Entrega 4 – Pagos | 31 marzo | 1 abril | | 25% | |
| Entrega 5 – Asignación académica | 21 abril | 22 abril | | 50% | |
| **Tercer Corte** | **Grupo 600** | **Grupo 800** | **40%** | |
| ***Parcial*** | ***26-27 mayo*** | ***26-27 mayo*** | | ***20%*** | |
| ***Proyecto*** | |  | | ***80%*** | |
| Entrega 6 – Notas | 5 mayo | 6 mayo | | 25% | |
| Entrega 7 – Reportes | 19 mayo | 20 mayo | | 25% | |
| Presentación final | 26 mayo | 27 mayo | | 50% | |

|  |  |  |
| --- | --- | --- |
| **ACCIÓN** | **NOTA** | **FECHA** |
| Reporte notas primer corte | 1er 30 % | 9-10 marzo |
| Modificaciones primer corte |  | 11-12 marzo |
| Reporte notas segundo corte | 2do 30 % | 20-21 abril |
| Modificaciones segundo corte |  | 22-23 abril |
| Reporte notas tercer corte | 3er 40 % | 29-30 mayo |
| Modificaciones segundo corte |  | 1 junio |
| Cierre y entrega de planillas |  | 2 junio |

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.

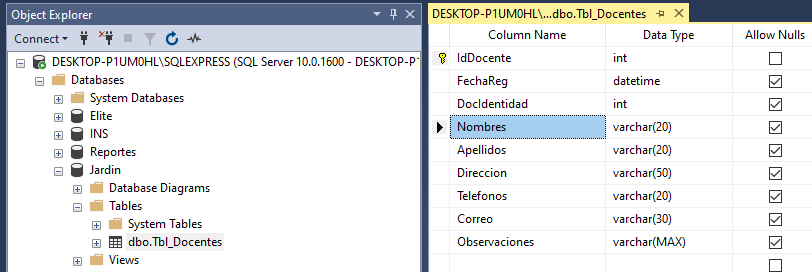
Para la entrega final el grupo deberá realizar la presentación de todo el proyecto, incluyendo la documentación.

Las entregas son acumuladas.

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

Se desarrollará el formulario para administrar Docentes, basado en la tabla Tbl\_Docentes de la base de datos Jardín

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 todos los Docentes, Buscar Docente por Id, Insertar, Actualizar y Eliminar.



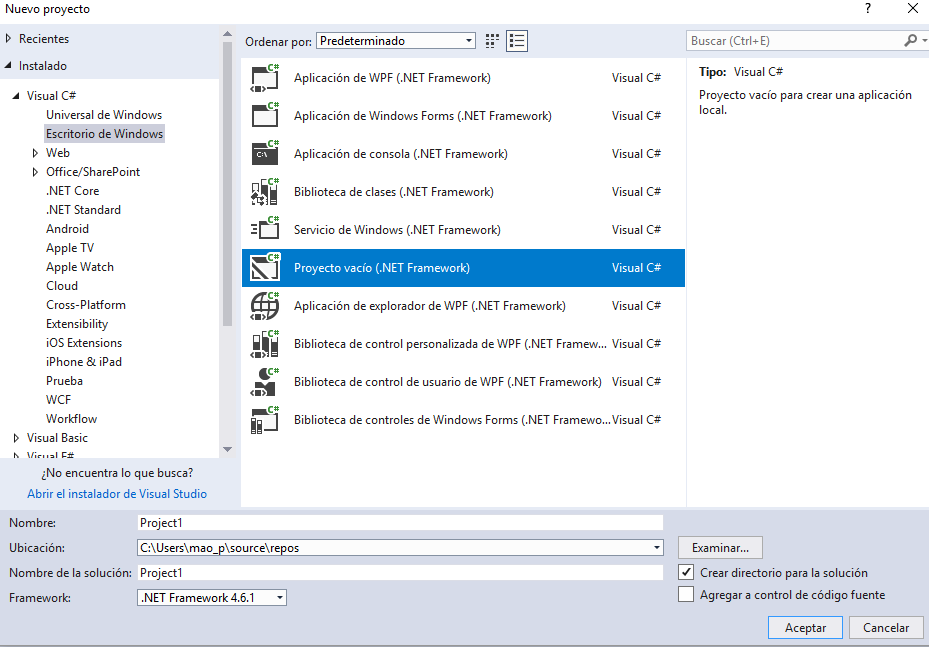
|  |  |
| --- | --- |
| USE Jardin  GO  --\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  CREATE PROCEDURE ListarDocente  AS  SELECT IdDocente [Id],  ISNULL(DocIdentidad,'') [Cedula],  ISNULL(Nombres,'')[Nombres],  ISNULL(Apellidos,'')[Apellidos],  ISNULL(Direccion,'')[Direccion],  ISNULL(Telefonos,'')[Telefonos],  ISNULL(Correo,'')[Email],  ISNULL(Observaciones,'')[Observaciones]  FROM Tbl\_Docentes  GO  --\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  CREATE PROCEDURE TraerDocentePorId  @ID int  AS  SELECT IdDocente [Id],  ISNULL(DocIdentidad,'') [Cedula],  ISNULL(Nombres,'')[Nombres],  ISNULL(Apellidos,'')[Apellidos],  ISNULL(Direccion,'')[Direccion],  ISNULL(Telefonos,'')[Telefonos],  ISNULL(Correo,'')[Email],  ISNULL(Observaciones,'')[Observaciones]  FROM Tbl\_Docentes  WHERE IdDocente = @ID  GO    --\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  CREATE PROCEDURE InsertarDocente  @DocIdentidad INT,  @Nombres VARCHAR(20),  @Apellidos VARCHAR(20),  @Direccion VARCHAR(50),  @Telefonos VARCHAR(20),  @Correo VARCHAR(30),  @Observaciones TEXT  AS  INSERT INTO Tbl\_Docentes(DocIdentidad,Nombres,Apellidos,  Direccion,Telefonos,Correo,Observaciones)  VALUES(@DocIdentidad,@Nombres,@Apellidos,  @Direccion,@Telefonos,@Correo,@Observaciones)  GO    --\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  CREATE PROCEDURE ActualizarDocente  @ID INT,  @DocIdentidad INT,  @Nombres VARCHAR(20),  @Apellidos VARCHAR(20),  @Direccion VARCHAR(50),  @Telefonos VARCHAR(20),  @Correo VARCHAR(30),  @Observaciones TEXT  AS  UPDATE Tbl\_Docentes SET  DocIdentidad=@DocIdentidad,  Nombres=@Nombres,  Apellidos=@Apellidos,  Direccion=@Direccion,  Telefonos=@Telefonos,  Correo=@Correo,  Observaciones=@Observaciones  WHERE IdDocente = @ID  GO    --\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  CREATE PROCEDURE EliminarDocente  @ID INT  AS  DELETE FROM Tbl\_Docentes  WHERE IdDocente = @ID  GO |  |

Ahora, vamos a construir el administrador de docentes

1. Crear una solución en blanco

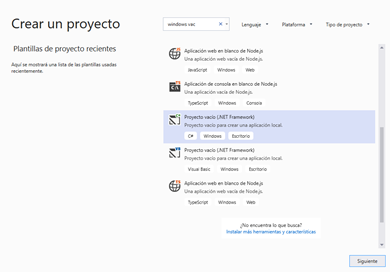
VISUAL STUDIO – 2012

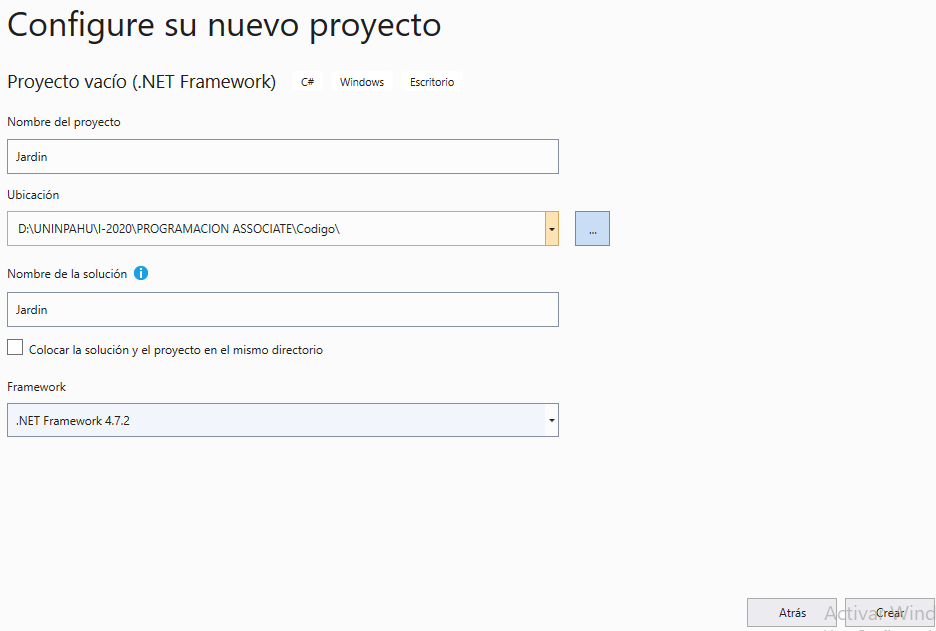
* Ingrese al menú Archivo => Nuevo => Proyecto
* Visual C# => Escritorio de Windows =>Proyecto Vacío
* Nombre de la solución Jardin\_xx
* Seleccione la ubicación del proyecto

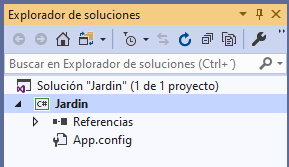


VISUAL STUDIO – 2019

* Busque proyecto vacío
* Nombre de la solución Jardin\_xx
* Seleccione la ubicación del proyecto





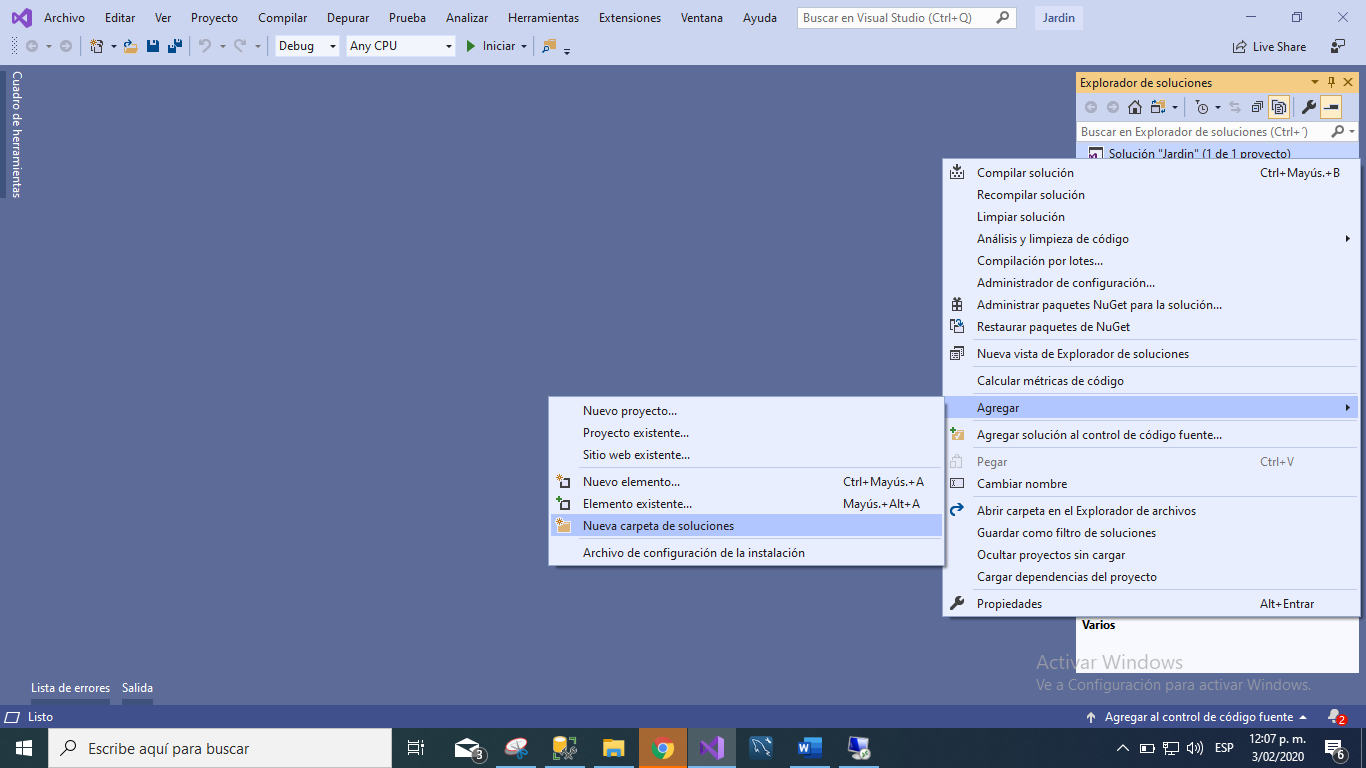


Recuerde que la arquitectura que se utilizara es MVC (Modelo-Vista-Controlador); para esto se crearan cada una de las capas y se conectaran una a una.

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

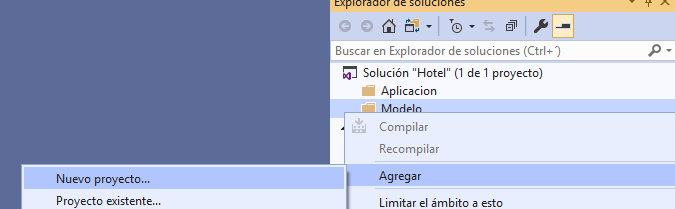
Clic derecho sobre la solución => agregar => nueva carpeta de soluciones

Asignar el nombre Modelo\_Controlador

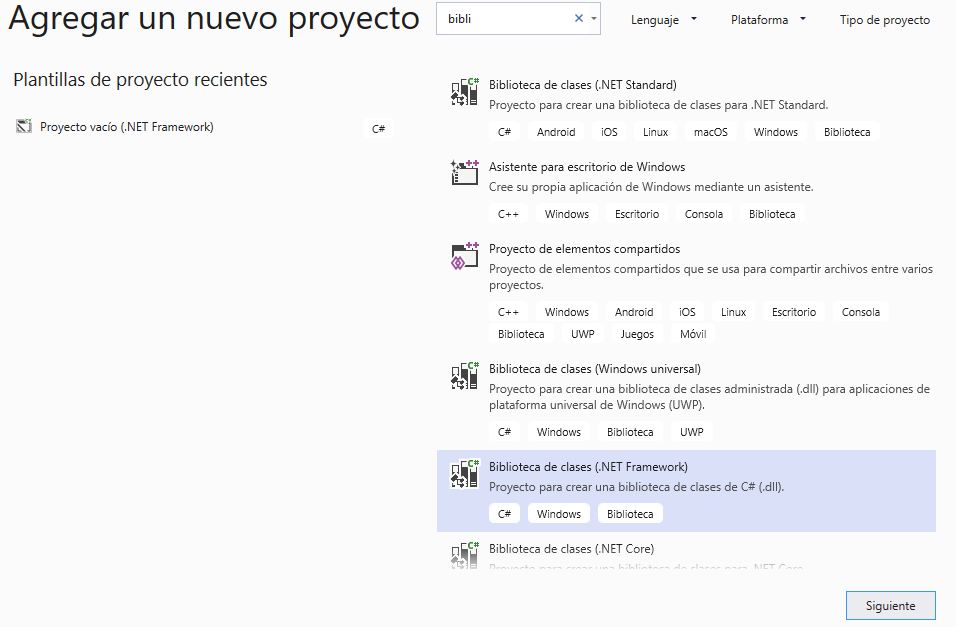


|  |  |
| --- | --- |
| Clic derecho sobre la solución => agregar => nueva carpeta de soluciones  Asignar el nombre Aplicaciones |  |

1. Sobre la Carpeta Modelo\_Controlador se crearán las capas de modelo (Entidad y Datos) y controlador (Negocio), clic derecho => agregar => nuevo proyecto



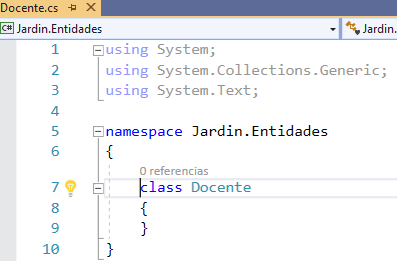
Busque => biblioteca de clases como tipo de proyecto. Asigne el nombre Jardin.Entidades



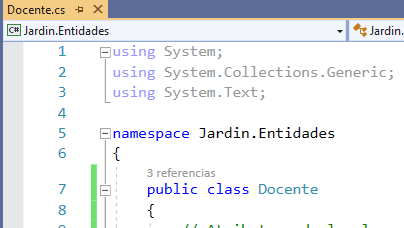


|  |  |
| --- | --- |
|  | Elimine la clase que se crea por defecto (Class1.cs)  Agregue un nuevo Item al proyecto, clic derecho => agregar => nuevo elemento |

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



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



Ahora se incluye el código dentro de las etiquetas por default de la clase:

using System;

using System.Collections.Generic;

using System.Text;

namespace Jardin.Entidades

{

public class Docente

{

// Atributos de la clase Docente

int \_id;

int \_DocIdentidad;

string \_Nombres;

string \_Apellidos;

string \_Direccion;

string \_Telefonos;

string \_Correo;

string \_Observaciones;

// Se definen los metodos get y set

public int Id

{

get { return \_id; }

set { \_id = value; }

}

public int DocIdentidad

{

get { return \_DocIdentidad; }

set { \_DocIdentidad = value; }

}

public string Nombres

{

get { return \_Nombres; }

set { \_Nombres = value; }

}

public string Apellidos

{

get { return \_Apellidos; }

set { \_Apellidos = value; }

}

public string Direccion

{

get { return \_Direccion; }

set { \_Direccion = value; }

}

public string Telefonos

{

get { return \_Telefonos; }

set { \_Telefonos = value; }

}

public string Correo

{

get { return \_Correo; }

set { \_Correo = value; }

}

public string Observaciones

{

get { return \_Observaciones; }

set { \_Observaciones = value; }

}

//Creamos los constructores de la clase. Se usa la sobrecarga

public Docente(int id, int DocIdentidad, string Nombres, string Apellidos, string Direccion, string Telefonos, string Correo, string Observaciones)

{

\_id = id;

\_DocIdentidad = DocIdentidad;

\_Nombres = Nombres;

\_Apellidos = Apellidos;

\_Direccion = Direccion;

\_Telefonos = Telefonos;

\_Correo = Correo;

\_Observaciones = Observaciones;

}

public Docente(int id, int DocIdentidad, string Nombres, string Apellidos)

{

\_id = id;

\_DocIdentidad = DocIdentidad;

\_Nombres = Nombres;

\_Apellidos = Apellidos;

}

public Docente() : this(0, 0, "", "", "", "", "", "")

{

}

}

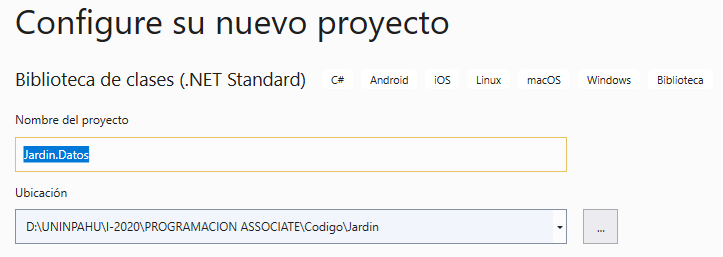
}

1. Crear la capa de Datos.

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

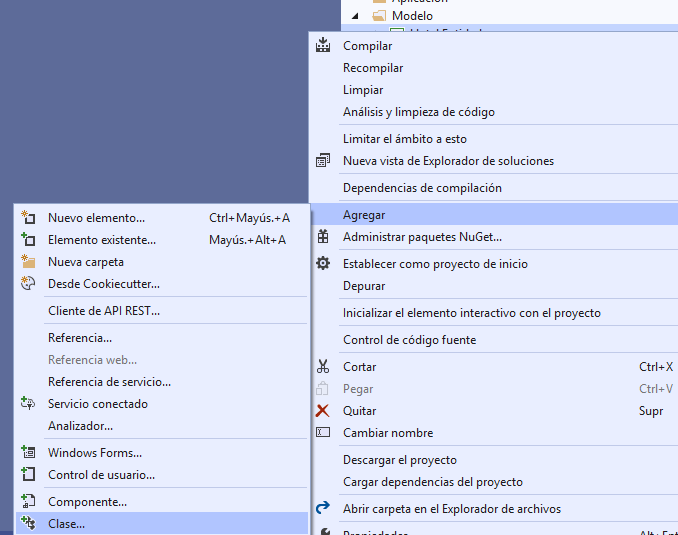
Seleccione Escritorio de Windows => biblioteca de clases como tipo de proyecto.

Asigne el nombre Jardin.Datos

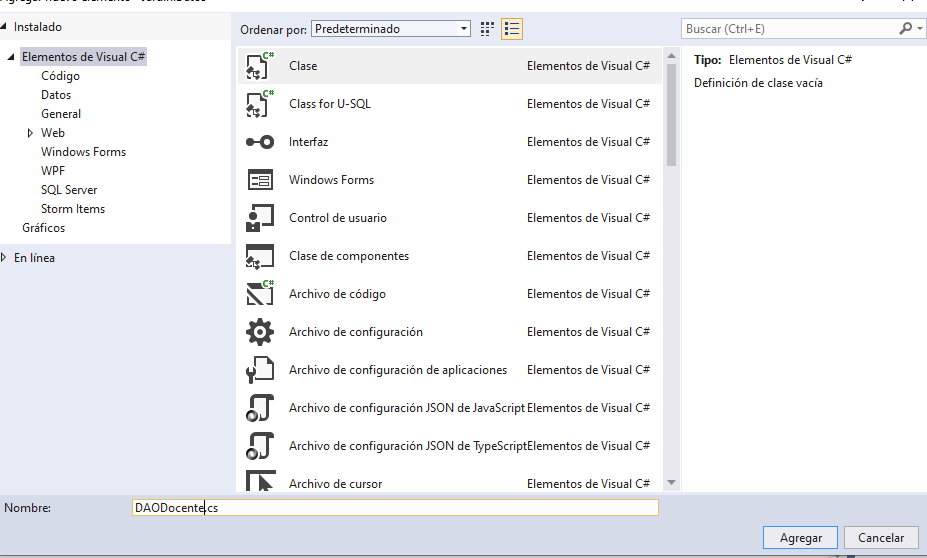


Elimine la clase que se crea por defecto (Class1.cs)

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



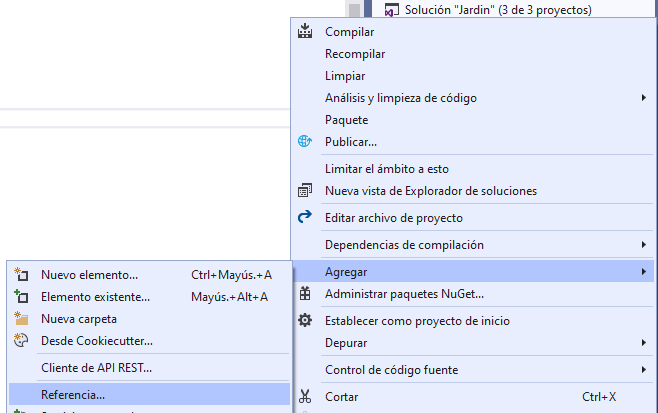
Seleccione el tipo Class y asigne el nombre DAODocente



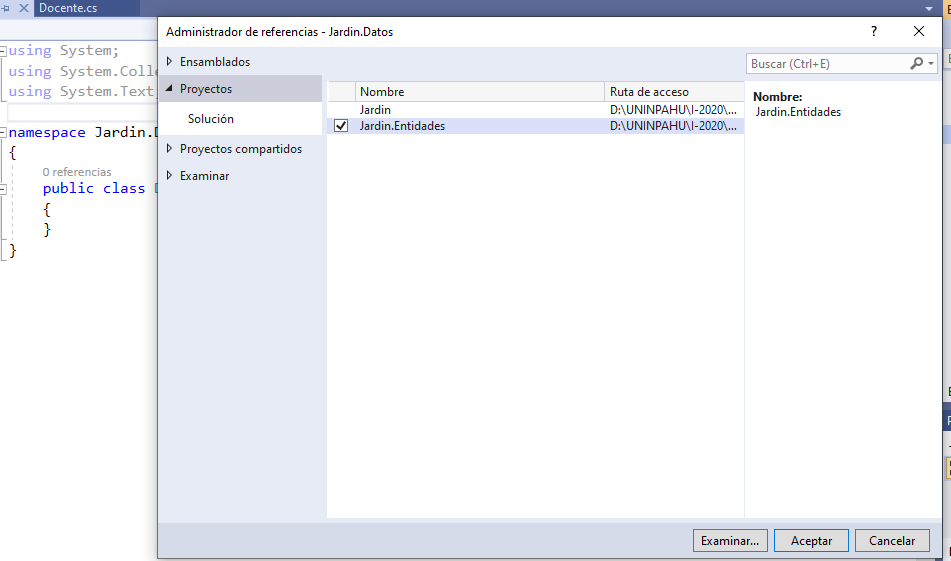
Para que la clase sea visible para los demás proyectos le agregamos el modificador public

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

Agregue la referencia al proyecto: ahora para que la capa de Datos lea la capa Entidades se debe adicionar la Referencia. Sobre el proyecto Jardin.Datos, hacer clic derecho => Agregar => Referencia,



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

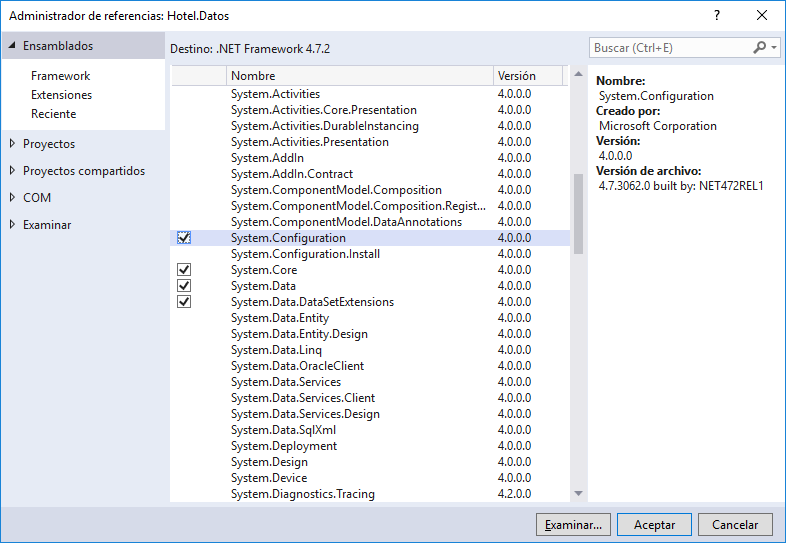


|  |  |
| --- | --- |
| Ya esta realizada la referencia entre la capa Datos con la capa Entidades; ahora se debe importar la librería dentro de la clase Datos para pode ser usada desde allí.  Usamos la instrucción siguiente en la parte superior donde se definen las librerías.  using Jardin.Entidades; |  |
|  | |

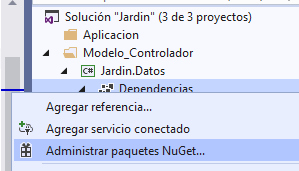
|  |  |
| --- | --- |
|  |  |

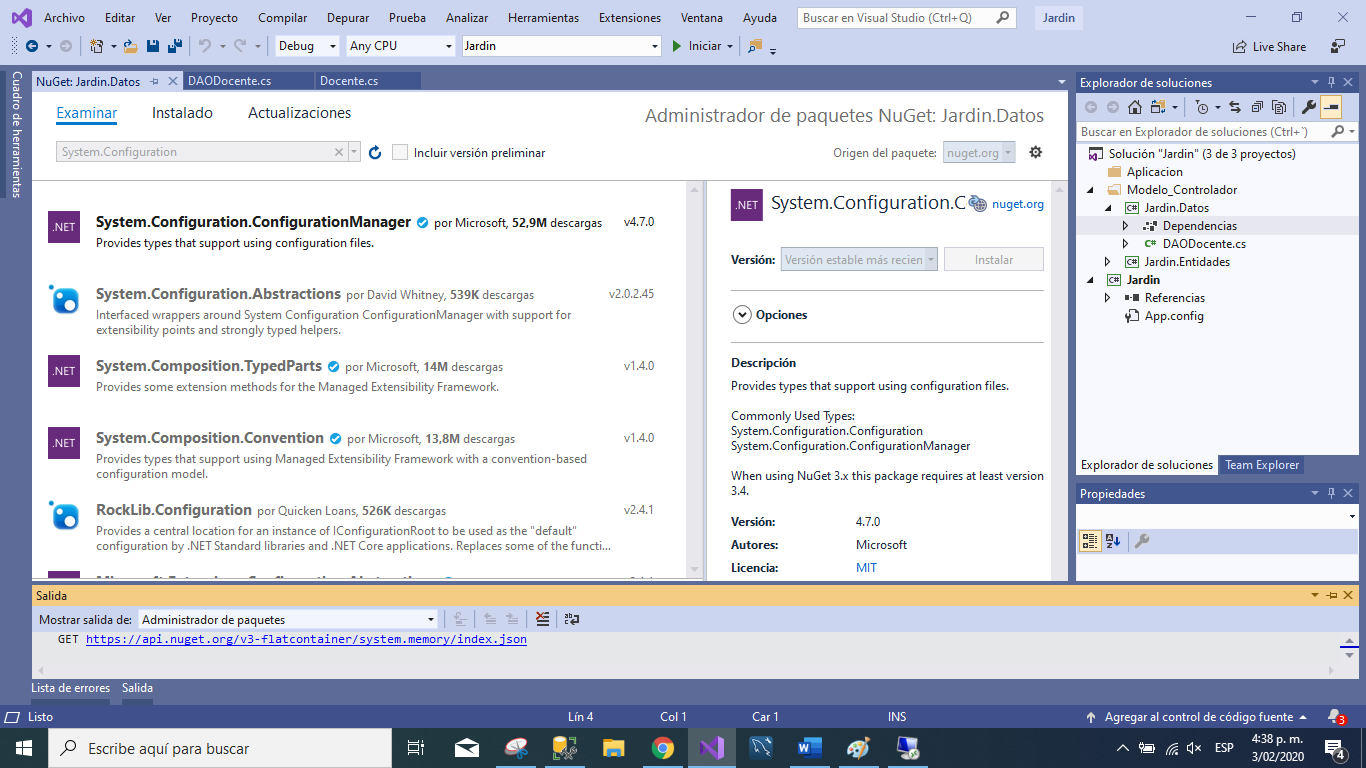
En esta capa también se debe referenciar la librería necesaria para establecer la conexión a la base de datos, que para este caso es SQLServer; así posteriormente poder crear la cadena de conexión en el AppConfig como en este caso que es Windows o WebConfig para Web.

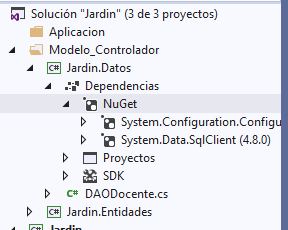
Ahora hacemos sobre el Proyecto Jardin.Datos, clic derecho => agregar referencia, seleccione la ficha Ensamblados, luego ubique System.Configuration y Aceptar. (esto solo se hace una vez)



Si no se encuentran la referencia, para la versión 2019, se instala el NuGet. Haciendo clic derecho, Administrar paquetes NuGet, se busca el requerido e Instalar. Deberá buscar System.Data.SqlClient y System.Configuration







Ahora sobre la clase DAODocente, 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. Esto con el fin de ser reconocias las instrucciones de Base de Datos y SQLServer.

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

Dentro de esta clase, 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 dentro de la clase DAODocente, debajo de la definición de la conexión, creamos los métodos de acceso a datos de esta entidad, tales como Listar, Insertar, Eliminar, Actualizar. Sugerencia crear una a una para validar su correcta codificación.

public List<Docente> Listar()

{

//Metodo que carga en una lista el resultado de todos los registros

de la tabla

List<Docente> lista = new List<Docente>();

using (SqlConnection con = new SqlConnection(CadenaConexion))

{

con.Open();

SqlCommand cmd = new SqlCommand("ListarDocente", con);

cmd.CommandType = CommandType.StoredProcedure;

SqlDataReader dr = cmd.ExecuteReader();

if (dr != null && dr.HasRows)

{

//Al definir los campos de la lista deben ser iguales a los

campos definidos en el

//procedimiento almacenado ListarDocente

while (dr.Read())

{

Docente c = new Docente((int)dr["Id"],

(int)dr["Cedula"], (string)dr["Nombres"],

(string)dr["Apellidos"], (string)dr["Direccion"],

(string)dr["Telefonos"], (string)dr["Email"],

(string)dr["Observaciones"]);

lista.Add(c);

}

}

}

return lista;

}

public Docente TraerPorId(int Id)

{

//Metodo que carga en un objeto el resultado del registro filtrado

por ID de la tabla

Docente oDocente = new Docente();

using (SqlConnection con = new SqlConnection(CadenaConexion))

{

con.Open();

SqlCommand cmd = new SqlCommand("TraerDocentePorId", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@ID", Id);

SqlDataReader dr = cmd.ExecuteReader();

if (dr != null && dr.HasRows)

{

dr.Read();

oDocente = new Docente((int)dr["Id"],

(int)dr["Cedula"], (string)dr["Nombres"],

(string)dr["Apellidos"], (string)dr["Direccion"],

(string)dr["Telefonos"], (string)dr["Email"],

(string)dr["Observaciones"]);

}

}

return oDocente;

}

public int Insertar(Docente oDocente)

{

// Metodo que permite enviar los datos del Insert a la BD

int n = -1;

using (SqlConnection con = new SqlConnection(CadenaConexion))

{

con.Open();

SqlCommand cmd = new SqlCommand("InsertarDocente", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@DocIdentidad", oDocente.DocIdentidad);

cmd.Parameters.AddWithValue("@Nombres", oDocente.Nombres);

cmd.Parameters.AddWithValue("@Apellidos", oDocente.Apellidos);

cmd.Parameters.AddWithValue("@Direccion", oDocente.Direccion);

cmd.Parameters.AddWithValue("@Telefonos", oDocente.Telefonos);

cmd.Parameters.AddWithValue("@Correo", oDocente.Correo);

cmd.Parameters.AddWithValue("@Observaciones", oDocente.Observaciones);

n = cmd.ExecuteNonQuery();

}

return n;

}

public int Actualizar(Docente oDocente)

{

// Metodo que permite enviar los datos del Update a la BD

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", oDocente.Id);

cmd.Parameters.AddWithValue("@DocIdentidad", oDocente.DocIdentidad);

cmd.Parameters.AddWithValue("@Nombres", oDocente.Nombres);

cmd.Parameters.AddWithValue("@Apellidos", oDocente.Apellidos);

cmd.Parameters.AddWithValue("@Direccion", oDocente.Direccion);

cmd.Parameters.AddWithValue("@Telefonos", oDocente.Telefonos);

cmd.Parameters.AddWithValue("@Correo", oDocente.Correo);

cmd.Parameters.AddWithValue("@Observaciones", oDocente.Observaciones);

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("EliminarDocente", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@Id", Id);

n = cmd.ExecuteNonQuery();

}

return n;

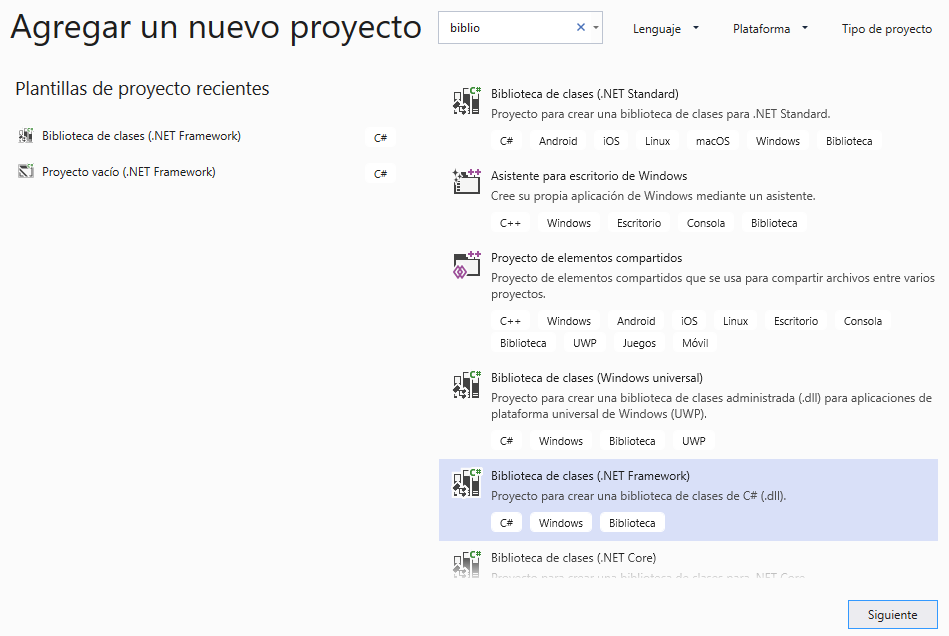
}

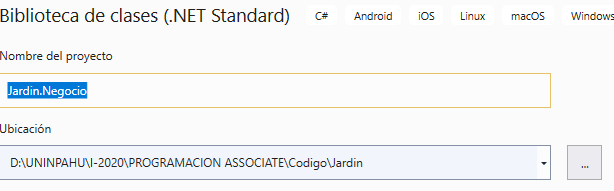
1. Crear la capa de Negocio.

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

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

Asigne el nombre Jardin.Negocio



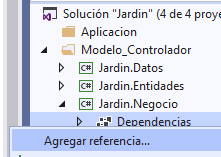


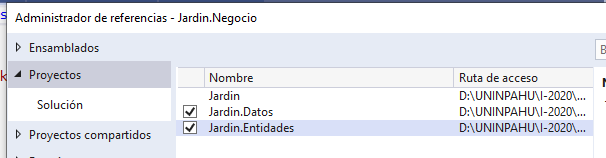
|  |  |
| --- | --- |
|  | Elimine la clase que se crea por defecto (Class1.cs)  Agregue un nuevo Item al proyecto, clic derecho => agregar => nuevo elemento  Seleccione el tipo Class y asigne el nombre BLDocente |

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

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

Agregue la referencia al proyecto Negocio.





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

Importamos la referencia para esto agregamos la siguiente instrucción.

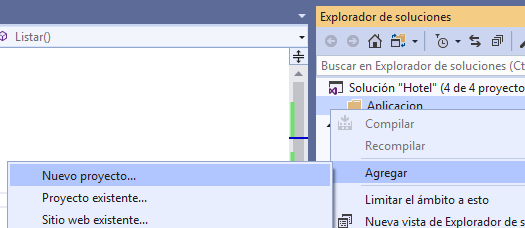
|  |  |
| --- | --- |
| using Jardin.Datos;  using Jardin.Entidades; |  |

En esta capa se crea la lógica del controlador, donde se relaciona el modelo con la vista invocando los métodos correspondientes. Para esto creamos los métodos tales como Listar, TraerPorId, Insertar, Eliminar, Actualizar.

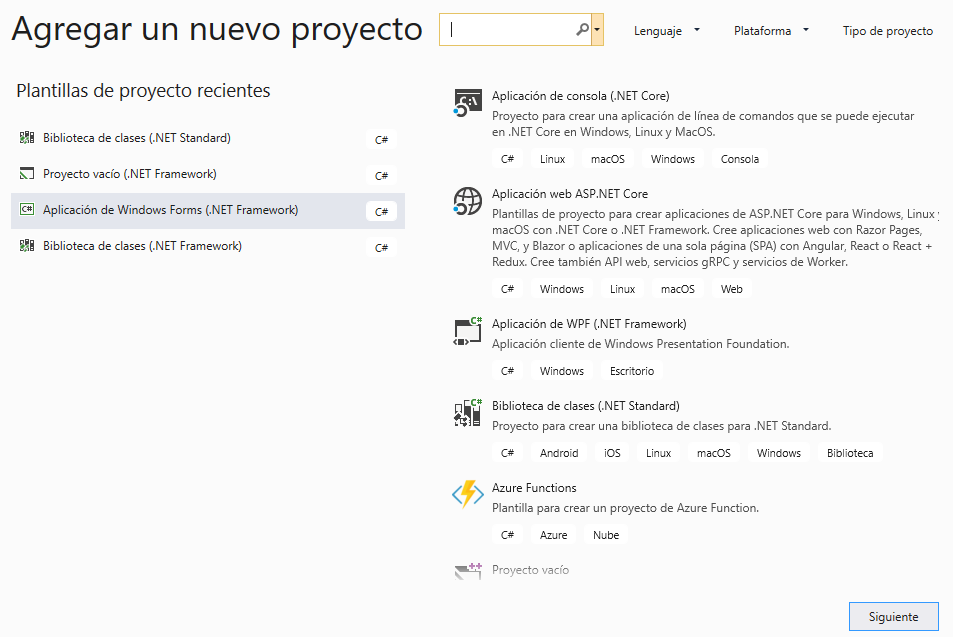
|  |  |
| --- | --- |
|  | public List<Docente> Listar()  {  DAODocente daDocente = new DAODocente();  return daDocente.Listar();  }  public Docente TraerPorId(int Id)  {  DAODocente daDocente = new DAODocente();  return daDocente.TraerPorId(Id);  }  public int Insertar(Docente oDocente)  {  DAODocente daDocente = new DAODocente();  return daDocente.Insertar(oDocente);  }  public int Actualizar(Docente oDocente)  {  DAODocente daDocente = new DAODocente();  return daDocente.Actualizar(oDocente);  }  public int Eliminar(int Id)  {  DAODocente daDocente = new DAODocente();  return daDocente.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.

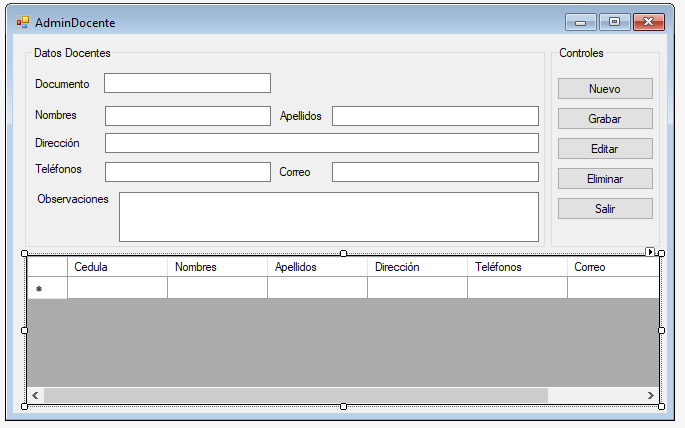


Asigne el nombre Jardin\_Windows

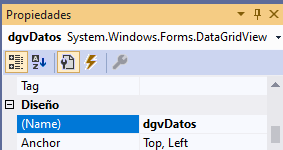
|  |  |
| --- | --- |
|  |  |

Aquí también debe referenciar los NuGet instalados anteriormente System.Data.SqlClient y System.Configuration

Diseñe el siguiente formulario: al formulario por default creado cámbiele el nombre o elimine y cree uno de nombre AdminDocente.



Utilizar las propiedades para asignar los nombres a los controles, por ejemplo:



En el GroupBox gbDatos: Contiene a todos los label y TextBox (txtDocumento; txtNombres; txtApellidos; txtDireccion; txtTelefonos; txtCorreo; txtObservaciones); el txtObservaciones debería mostrarse multilinea

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

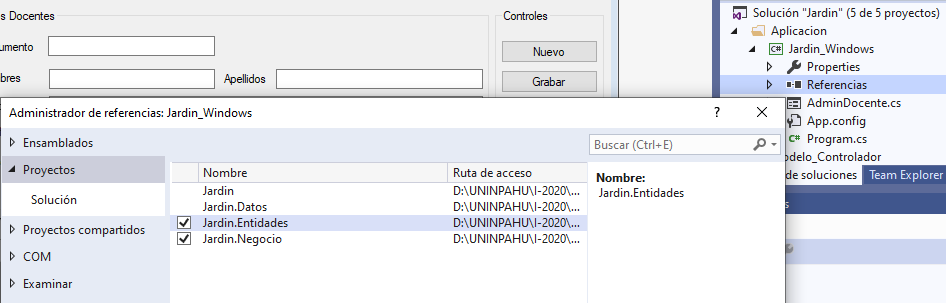
**DataGridView** dgvDatos; Personalize el control dgvDatos, agregue las columnas Id, Cedula, Nombres, Apellidos, Dirección, Teléfono, Correo, Observación. Modifique la propiedad Visible=False de la columna Id.

|  |  |
| --- | --- |
|  | Sobre la flecha hacer clic derecho    Seleccionar Editar columnas.. |

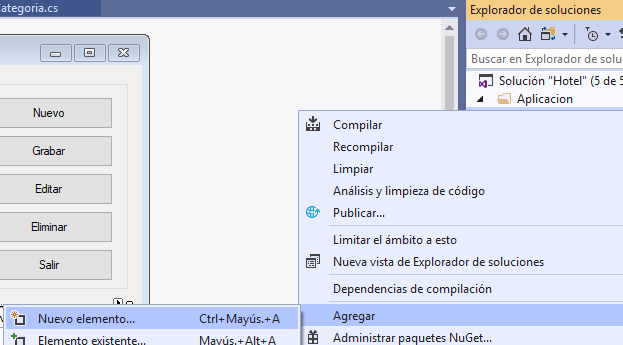
|  |  |
| --- | --- |
|  |  |

Agregue la referencia al proyecto:

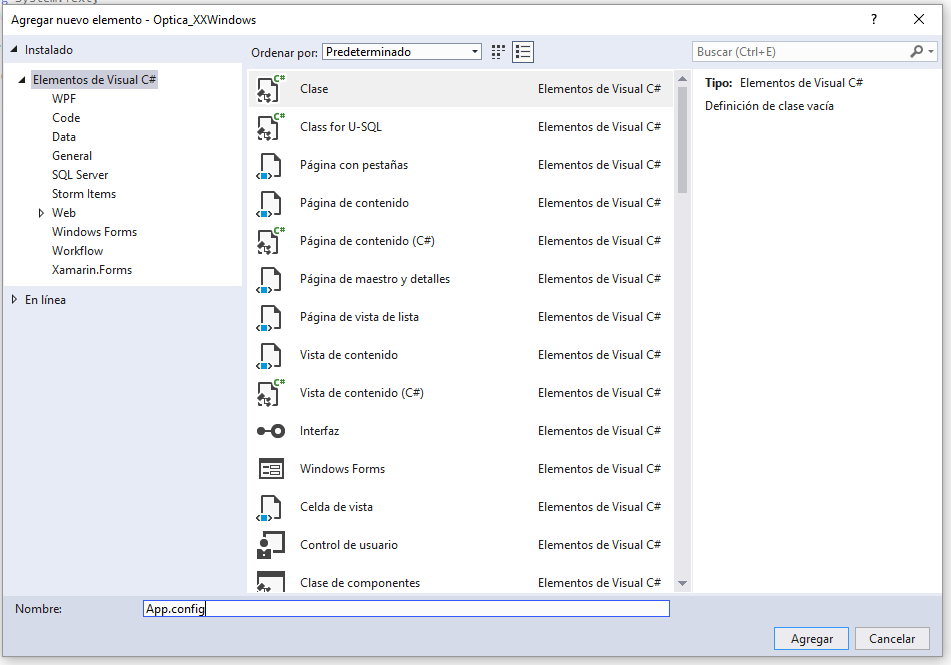
Proyecto Jardin\_WEindows, clic derecho => Agregar =>Referencia, Seleccione la ficha Proyectos, luego seleccione el proyecto Jardin.Entidades y Jardin.Negocio => Aceptar

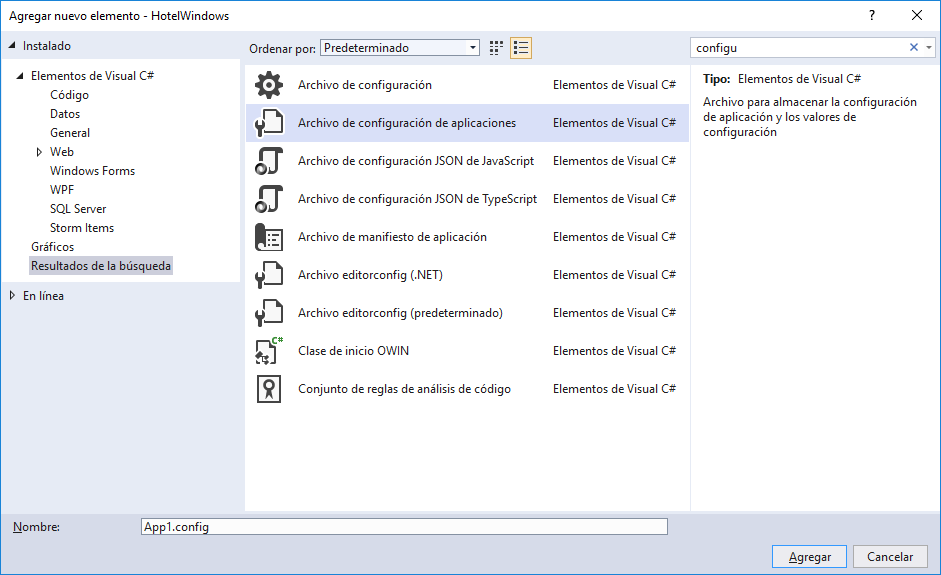


Agregue un nuevo Item al proyecto, clic derecho => Agregar => nuevo elemento



Seleccione el tipo Application Configuration File, el nombre será por defecto App.config. Cuando se crea el proyecto este archivo se inicializa por defecto también.





Abrimos el archivo App.config y agregamos la siguiente instrucción después de <configuration>, para definir la cadena de conexión a la BD, se puede usar con la seguridad explicita Conex1 o la seguridad integrada Conex

<connectionStrings>

<add name="Conex" connectionString="Data Source= NOMBRE\_SERVIDOR;

Initial Catalog= NOMBRE\_BD; Integrated Security=true"/>

</connectionStrings>

<connectionStrings>

<add name="Conex1" connectionString="Data Source=NOMBRE\_SERVIDOR;

Initial Catalog= NOMBRE\_BD; UID = usuarioBD; PWD = ClaveBD"/>

</connectionStrings>

Ahora, dentro del formulario hacemos clic derecho y seleccionamos **Ver Código**. Allí se abre el código para el formulario donde, Importamos la referencia para esto agregamos la siguiente instrucción.

|  |  |
| --- | --- |
| using Jardin.Entidades;  using Jardin.Negocio; |  |

Declaramos las variables generales, dentro de la clase principal que hereda el formulario

|  |  |
| --- | --- |
| //Variables generales  List<Docente> lista = null;  BLDocente blDocente = new BLDocente();  Docente c;  bool \_nuevo = false; |  |
| Dentro de la clase principal, luego de la inicialización de los componentes del formulario se invocan los métodos:  ActivarControlDatos(gbDatos, false);  CargarDatos();  Inicialmente se vera como error por que no se han creado los métodos. |  |

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 = blDocente.Listar();  }  if (lista.Count > 0)  {  dgvDatos.Rows.Clear();  for (int i = 0; i < lista.Count; i++)  {  dgvDatos.Rows.Add(lista[i].Id,  lista[i].DocIdentidad,  lista[i].Nombres,  lista[i].Apellidos,  lista[i].Direccion,  lista[i].Telefonos,  lista[i].Correo,  lista[i].Observaciones);  }  }  } |  |

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 Docente(0, int.Parse(txtDocumento.Text),  txtNombres.Text, txtApellidos.Text,  txtDireccion.Text, txtTelefono.Text,  txtCorreo.Text, txtObservaciones.Text);  n = blDocente.Insertar(c);  }  else  {  c.DocIdentidad = int.Parse(txtDocumento.Text);  c.Nombres = txtNombres.Text;  c.Apellidos = txtApellidos.Text;  c.Direccion = txtDireccion.Text;  c.Telefonos = txtTelefono.Text;  c.Correo = txtCorreo.Text;  c.Observaciones = txtObservaciones.Text;  n = blDocente.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 = blDocente.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 = blDocente.TraerPorId((int)dgvDatos[0, dgvDatos.CurrentRow.Index].Value);

txtDocumento.Text = c.DocIdentidad.ToString();

txtNombres.Text = c.Nombres;

txtApellidos.Text = c.Apellidos;

txtDireccion.Text = c.Direccion;

txtTelefono.Text = c.Telefonos;

txtCorreo.Text = c.Correo;

txtObservaciones.Text = c.Observaciones;

ActivarControlDatos(gbDatos, true);

ActivarButton(false);

dgvDatos.Enabled = false;

btnEditar.Text = "Cancelar";

}

}

}

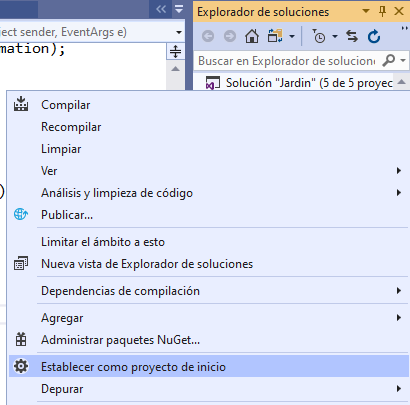
Botón Eliminar

|  |  |
| --- | --- |
|  | private void btnEliminar\_Click(object sender, EventArgs e)  {  if (dgvDatos.RowCount > 0)  {  c = blDocente.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 = blDocente.Eliminar(c.Id);  if (n > 0)  {  MessageBox.Show("Registro eliminado", "Aviso",  MessageBoxButtons.OK, MessageBoxIcon.Information);  lista = blDocente.Listar();  CargarDatos();  }  else  {  MessageBox.Show("Error al eliminar", "Aviso",  MessageBoxButtons.OK, MessageBoxIcon.Error);  }  }  }  } |

Botón Salir

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

Para Finalizar haga clic derecho sobre el proyecto Jardin\_Windows y selecciones establecer como proyecto principal.



Puede quitar el proyecto inicial del default, haciendo clic derecho sobre el proyecto y seleccionando quitar.

|  |  |
| --- | --- |
| Proyecto default inicial |  |
| Como debe quedar la solución |

Ahora puede ejecutar la Aplicación

