Universidad Mariano Gálvez de Guatemala

Escuela Nacional Central de Agricultura

–ENCA-

Facultad de Ingeniería en Sistemas y Ciencias de la Computación Ejercicio Profesional Supervisado –EPS-

Informe de Resultados para la Escuela Nacional Central de Agricultura –ENCA-, Bajo Subvención y Programación de Desembolsos.

Periodo: DICIEMBRE-2023

f. _____

EPSista. Briam Garcia Gómez.

f. _____

Ing. Gloria Ercira Colindres Solórzano.

Encargada de la unidad de Informática.

f. _____

Ing. Ronny Estuardo Mancilla Ruano, M.Sc.

Director ENCA.

30-12-2023

Informe Avances Sistema Recursos Humanos

EPSista. Briam Garcia Gómez

Índice

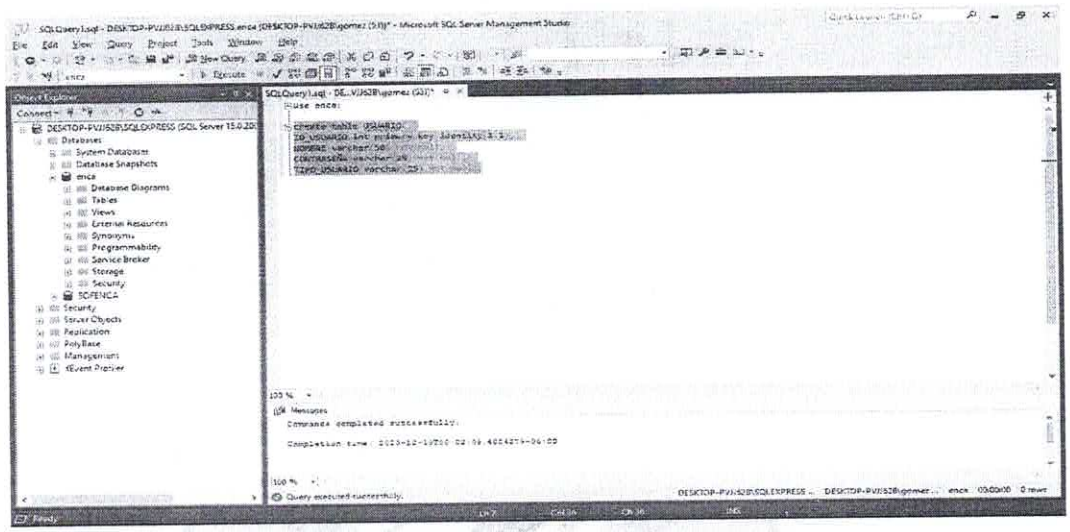
Introducción.

En el desarrollo de aplicaciones, la gestión eficiente de datos es esencial. En este contexto, SQL Server se destaca como una potente plataforma de gestión de bases de datos. En este avance, exploraremos el proceso de creación de tablas en SQL Server, la inserción de datos en esas tablas y la posterior implementación de procedimientos almacenados. Además, abordaremos la conexión de la base de datos a un proyecto en Windows Forms utilizando como lenguaje de programación C#. A medida que avanzamos, nos enfocaremos en la estructuración de la aplicación mediante la arquitectura de capas, distinguiendo entre capa de datos, capa de negocio y capa de vista de usuario.

Avance de creación de tablas y insert de datos a las tablas.

Creación de avances de tablas.

Vamos a nuestra base de datos. En SQL Server y nos conectamos a nuestra base de datos.



Ya conectados a nuestro servidor crearemos las tablas y sus insert.

Comandos sql o script.

create table USUARIO(ID_USUARIO int primary key identity(1,1), NOMBRE varchar(50) not null, CONTRASEÑA varchar(25) not null, TIPO_USUARIO varchar(25) not null);

create table TYRENGLONPRESUPUESTARIO(ID_RENGLON int primary key identity(1,1), RENGLON_PRESUPUESTARIO varchar(50) not null, ABREVIATURA varchar(5) not null, FLAG_BONOFICACION int not null, FLAG_IGSS int not null,

FLAG_AGUINALDO int not null, FLAG_BONO14 int not null, FLAG_MONTEPIO int not null);

create table TYESTADOCIVIL(ID_ESTADO_CIVIL int primary key identity(1,1), ESTADO_CIVIL varchar(30) not null);

Create table TYGENERO(ID_GENERO int primary key identity(1,1), GENERO varchar(15) not null);

create table TYETNIA(ID_ETNIA int primary key identity(1,1), ETNIA varchar(40) not null);

create table TYRELIGION(ID_RELIGION int primary key identity(1,1), RELIGION varchar(40));

create table TYPUESTONOMINAL(ID_PUESTO_NOMINAL int primary key identity(1,1), PUESTO_NOMINAL varchar(70));

create table TYDEPARTAMENTO(ID_DEPARTAMENTO int primary key identity(1,1), DEPARTAMENTO varchar(30) not null, CODIGO_POSTAL varchar(5) not null);

create table TYMUNICIPIO(ID_MUNICIPIO int primary key identity(1,1), ID_DEPARTAMENTO int, MUNICIPIO varchar(100) not null, CODIGO_POSTAL varchar(5)not null, CONSTRAINT FK_TyMunicipio_TyDepartamento FOREIGN KEY (Id_Departamento) REFERENCES TyDepartamento(Id_Departamento));

create table TYCOORDINACION(ID_COORDINACION int primary key identity(1,1), COORDINACION varchar(100) not null);

```
create table TYUNIDADSECCION( IDUNIDAD_SECCION int primary key
identity(1,1), ID_COORDINACION int, UNIDAD_SECCION varchar(100) not null,
CONSTRAINT FK_TyCoordinacion_TyUnidadSeccion FOREIGN KEY
(Id_Coordinacion) REFERENCES TyCoordinacion(Id_Coordinacion));

--insertando en la tabla TyCoordinacion INSERT INTO
TyCoordinacion(Coordinacion) VALUES('Dirección'),('Coordinación
Académica'),('Coordinación de Producción'), ('Coordinación de Vida
Estudiantil'),('CENAF'),('Coordinación Administrativa y Financiera');

INSERT INTO TyUnidadSeccion(Id_Coordinacion,Unidad_Seccion)
VALUES(1,'Sección de Personal'),(1,'Sección de Planificación
Institucional'),(1,'Sección Cooperación Externa'), (1,'Sección de
Investigación'),(1,'Asesoría Jurídica'),(1,'Unidad de Acceso a la Información
Pública y Comunicación Social '), (1,'Unidad de Informática'),(1,'Fincas ');

INSERT INTO TyUnidadSeccion(Id_Coordinacion,Unidad_Seccion)
VALUES(2,'Unidad de Control Académico'),(2,'Unidad de
Admisión'),(2,'Administración Educativa'),(2,'Catedráticos');

INSERT INTO TyUnidadSeccion(Id_Coordinacion,Unidad_Seccion)
VALUES(3,'Unidad de Comercialización e Industrialización'),(3,'Áreas de
Producción');

INSERT INTO TyUnidadSeccion(Id_Coordinacion,Unidad_Seccion)
VALUES(4,'Servicio de Salud'),(4,'Servicios Generales Estudiantes'),
(4,'Supervisión Estudiantil'),(4,'Cultura y Deporte');
```

INSERT INTO TyUnidadSeccion(Id_Coordinacion,Unidad_Seccion)

VALUES(5,'Estudios y Desarrollo Académico'),(5,'Operación y Gestión');

INSERT INTO TyUnidadSeccion(Id_Coordinacion,Unidad_Seccion)

VALUES(6,'Presupuesto'),(6,'Contabilidad'),

(6,'Inventarios'),(6,'Tesorería'),(6,'Sección de Compras'),(6,'Analistas de Compra'),

(6,'Sección Administrativo'),(6,'Almacén'),(6,'Unidad de Servicios Generales');

-- Insertar tipos de género

INSERT INTO TyGenero (Genero) VALUES('Masculino'),('Femenino');

-- Insertar religiones

INSERT INTO TyReligion (Religion)

VALUES('Católica'),('Evangélica'),('Testigos de Jehová'),('Mormones'),('Adventista del Séptimo Día'), ('Judaísmo'),('Islam'),('Budismo'),('Hinduismo'),('Ninguna');

--insert tabla Usuario

INSERT INTO Usuario(Nombre,Contraseña,Tipo_Usuario)

VALUES('Briam','%123$sys','Administrador');

Este usuario nos ayudara a configurar el login para avanzar en la aplicación.

INSERT INTO TyEstadoCivil (Estado_Civil) VALUES ('Soltero/a'),

('Casado/a'), ('Divorciado/a'), ('Viudo/a'), ('Separado/a'), ('Unión libre/convivencia');

INSERT INTO TyDepartamento (Departamento, Codigo_Postal) VALUES

('GUATEMALA', '01000'),('EL PROGRESO', '02000'),('SACATEPEQUEZ',

'03000'),('CHIMALTENANGO', '04000'), ('ESCUINTLA', '05000'),('SANTA ROSA',

'06000'),('SOLOLA', '07000'),('TOTONICAPAN', '08000'), ('QUETZALTENANGO', '09000'),('SUCHITEPEQUEZ', '10000'),('RETALHULEU', '11000'),('SAN MARCOS', '12000'), ('HUEHUETENANGO', '13000'),('QUICHE'; '14000'),('BAJA VERAPAZ', '15000'),('ALTA VERAPAZ', '16000'),('PETEN', '17000'), ('IZABAL', '18000'),('ZACAPA', '19000'),('CHIQUIMULA', '20000'),('JALAPA', '21000'),('JUTIAPA', '22000');

Nota. Al insertar en la tabla Municipio se hace un insert lógico que corresponde a cada departamento con sus municipios. Ejemplo.

-- Insertar municipios del departamento GUATEMALA INSERT INTO TyMunicipio (Id_Departamento, Municipio, Codigo_Postal) VALUES (1, 'SANTA CATARINA PINULA', '01051'), (1, 'SAN JOSE PINULA', '01052'), (1, 'SAN JOSE DEL GOLFO', '01053'), (1, 'PALENCIA', '01054'), (1, 'CHINAUTLA', '01055'), (1, 'SAN PEDRO AYAMPUC', '01056'), (1, 'MIXCO', '01057'), (1, 'SAN PEDRO SACATEPEQUEZ', '01058'), (1, 'SAN JUAN SACATEPEQUEZ', '01059'), (1, 'SAN RAYMUNDO', '01060'), (1, 'CHUARRANCHO', '01061'), (1, 'FRAIJANES', '01062'), (1, 'AMATITLAN', '01063'), (1, 'VILLA NUEVA', '01064'), (1, 'VILLA CANALES', '01065'), (1, 'PETAPA', '01066'), (1, 'CANALITOS (ZONA 21)', '01067'), (1, 'EL FISCAL (PALENCIA)', '01068'), (1, 'BOCA DEL MONTE (VILLA CANALES)', '01069'), (1, 'VUELTA GRANDE (SAN RAYMUNDO)', '01070'), (1, 'TRAPICHE GRANDE (CHUARRANCHO)', '01071'), (1, 'SANTA ELENA BARILLAS (VILLA CANALES)', '01072'), (1, 'PUERTA PARADA (SANTA CATARINA PINULA)', '01073'); se inserta con el Id de departamento y sus municipios y así en todos los departamentos.

```
create table TYVIVIENDA( IDVIVIENDA int primary key identity(1,1),
VIVIENDA VARCHAR(40) not null);

--insert tyvivienda

insert into TYVIVIENDA(VIVIENDA)
values('Propio'),('Familiar'),('Renta'),('Huéspedes');

create table TYIDIOMA( IDIDIOMA int primary key identity(1,1), IDIOMA
varchar(50) not null);

--insert tyidioma

insert into TYIDIOMA(IDIOMA)
values('Español'),('Inglés'),('Francés'),('Alemán'),('Italiano'),('Portugués'),
('Chino'),('Japonés'),('Ruso'),('Árabe');


create table TYVEHICULO( IDVEHICULO int primary key identity(1,1),
VEHICULO varchar(40));

--insert tyvehiculo

insert into TYVEHICULO(VEHICULO) values('Propio'),('Familiar'),('Lo está
pagando');

create table TYAGRUPACION( IDAGRUPACION int primary key
identity(1,1), AGRUPACION varchar(50));
```

--insert TYAGRUPACION insert into TYAGRUPACION(AGRUPACION) values('Asociación'),('Sindicato'),('Partido Político'),('Otro '),('No Procede');

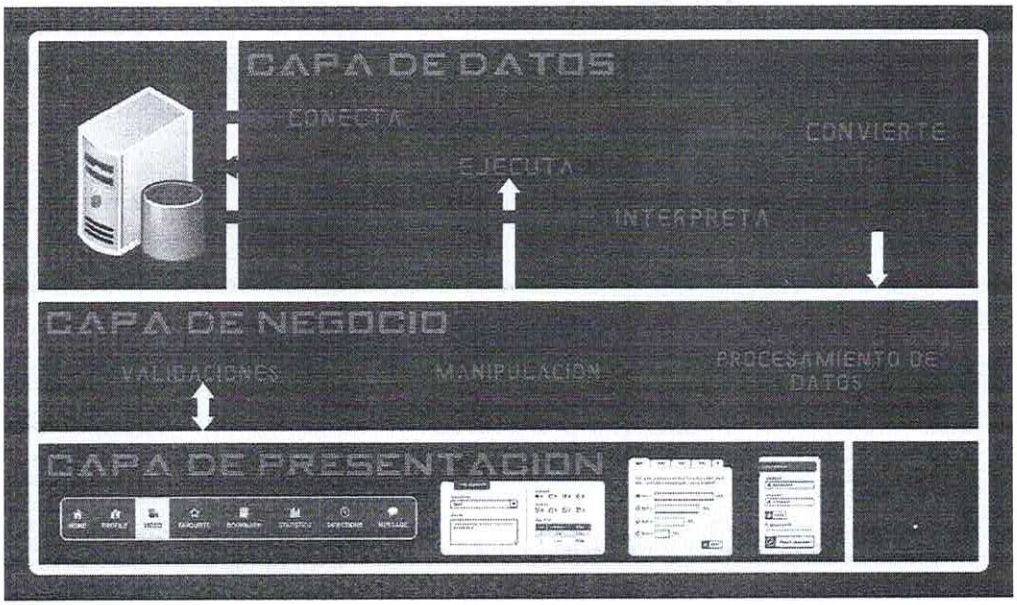
create table TYLOCALIZACION(IDLOCALIZACION int primary key identity(1,1), DESCRIPCION varchar(50), FLAG_OBLIGATORIO varchar(3));

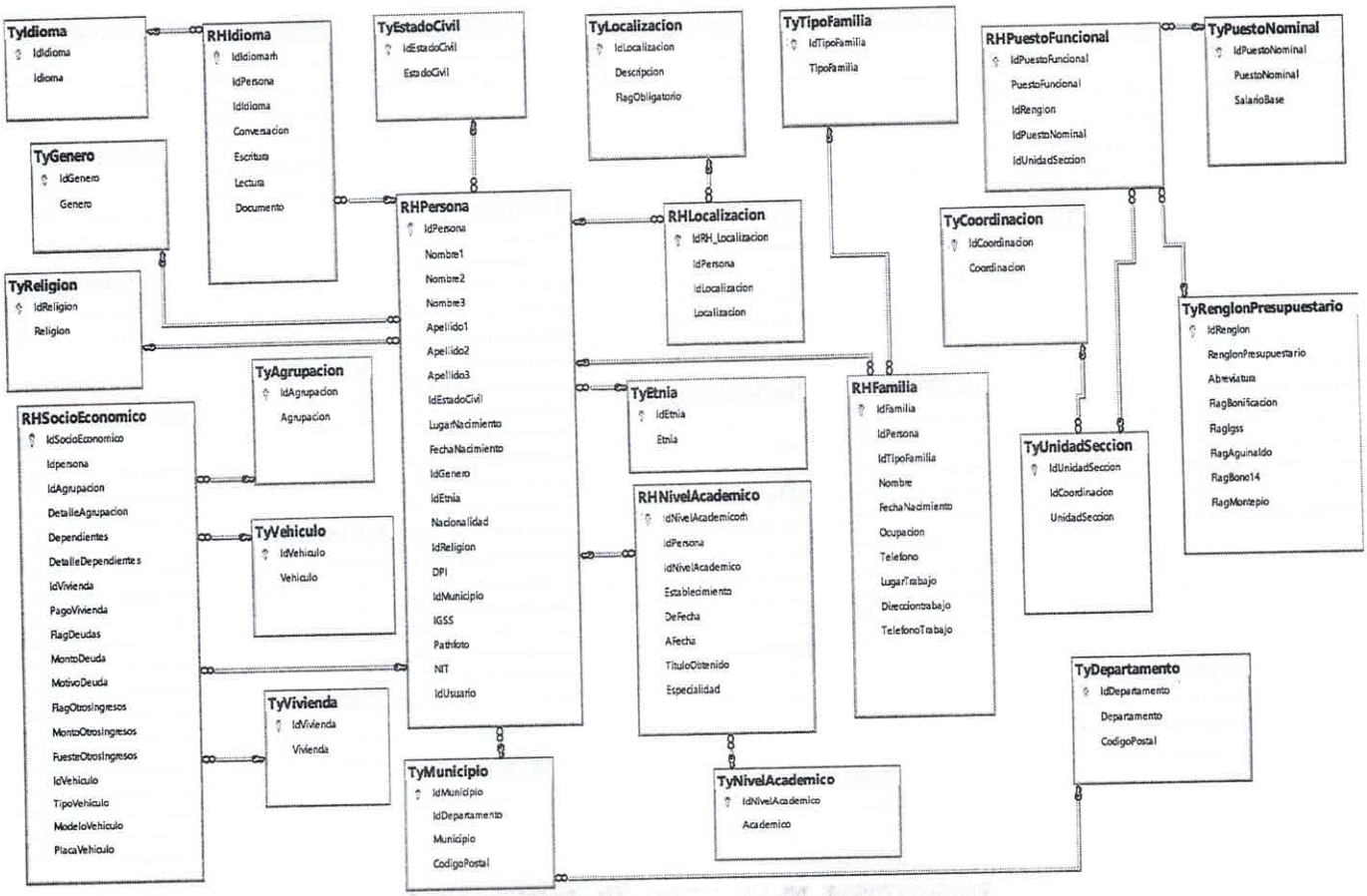
--insert tylocalizacion

insert into TYLOCALIZACION(DESCRIPCION,FLAG_OBLIGATORIO) values('Correo Electrónico','1'),('Dirección','1'),('Telefono','1'),('Telefono2','0'),('Otra','0');

con esto terminamos con el siguiente diagrama de nuestra base de datos.

La Aplicación se trabajará con la siguiente arquitectura.





Ya teniendo nuestras relaciones bien estructuradas de nuestra base de datos se crearán los procedimientos almacenadas de las tablas RHPersona, RHLocalizacion, RHFamilia, RHIdioma, RHNivelAcademico, RHPuestoFuncional, RHSocioEconomico.

Que son las tablas que nos ayudaran a darle lógica a nuestra aplicación.

Creación de procedimientos almacenados de INSERT, DELETE, SELECT y UPDATE de las tablas de Mantenimiento de la aplicación.

Procedimiento almacenado de listar datos RHPersona.

```
USE [encaagro]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER procedure [dbo].[sp_ListarPersonaIdioma]
as
begin
select IdPersona, DPI from RhPersona;
end;
```

Procedimiento almacenado de inserter RHPersona.

```
USE [encaagro]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[sp_InsertarPersona]
    @P_NOMBRE NVARCHAR(30),
    @S_NOMBRE NVARCHAR(30),
    @T_NOMBRE NVARCHAR(30),
    @P_APELLIDO NVARCHAR(30),
    @S_APELLIDO NVARCHAR(30),
    @C_APELLIDO NVARCHAR(30),
    @EDAD nvarchar(20),
    @PRETENCION_SALARIO nvarchar(20),
    @ID_ESTADO_CIVIL INT,
    @L_NACIMIENTO NVARCHAR(100),
    @F_NACIMIENTO DATE,
    @ID_GENERO INT,
    @ID_ETNIA INT,
    @NACIONALIDAD NVARCHAR(35),
    @ID_RELIGION INT,
    @DPI NVARCHAR(15),
    @ID_MUNICIPIO INT,
    @IGSS NVARCHAR(12),
    @NIT NVARCHAR(12),
    @Foto image,
    @LICENCIA NVARCHAR(10),
    @TIPO_LICENCIA NVARCHAR(10)
AS
BEGIN
    INSERT INTO RHPersona (
        P_NOMBRE, S_NOMBRE, T_NOMBRE, P_APELLIDO, S_APELLIDO,
        C_APELLIDO, EDAD, PRETENCION_SALARIO, ID_ESTADO_CIVIL,
        L_NACIMIENTO, F_NACIMIENTO, ID_GENERO, ID_ETNIA,
```

```
        NACIONALIDAD, ID_RELIGION, DPI, ID_MUNICIPIO,
        IGSS, NIT, Foto, LICENCIA, TIPO_LICENCIA
    )
    VALUES (
        @P_NOMBRE, @S_NOMBRE, @T_NOMBRE, @P_APELLIDO, @S_APELLIDO,
        @C_APELLIDO, @EDAD, @PRETENCION_SALARIO, @ID_ESTADO_CIVIL,
        @L_NACIMIENTO, @F_NACIMIENTO, @ID_GENERO, @ID_ETNIA,
        @NACIONALIDAD, @ID_RELIGION, @DPI, @ID_MUNICIPIO,
        @IGSS, @NIT, @Foto, @LICENCIA, @TIPO_LICENCIA
    );
END;
```

Procedimiento almacenado de Actualizar persona.

```
USE [encaagro]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[sp_ActualizarPersona]
    @IdPersona INT,
    @P_NOMBRE NVARCHAR(30),
    @S_NOMBRE NVARCHAR(30),
    @T_NOMBRE NVARCHAR(30),
    @P_APELLIDO NVARCHAR(30),
    @S_APELLIDO NVARCHAR(30),
    @C_APELLIDO NVARCHAR(30),
    @EDAD nvarchar(20),
    @PRETENCION_SALARIO nvarchar(20),
    @ID_ESTADO_CIVIL INT,
    @L_NACIMIENTO NVARCHAR(100),
    @F_NACIMIENTO DATE,
    @ID_GENERO INT,
    @ID_ETNIA INT,
    @NACIONALIDAD NVARCHAR(35),
    @ID_RELIGION INT,
    @DPI NVARCHAR(15),
    @ID_MUNICIPIO INT,
    @IGSS NVARCHAR(12),
    @NIT NVARCHAR(12),
    @Foto image,
    @LICENCIA NVARCHAR(10),
    @TIPO_LICENCIA NVARCHAR(10)
AS
BEGIN
    UPDATE RHPersona
    SET
        P_NOMBRE = @P_NOMBRE,
        S_NOMBRE = @S_NOMBRE,
        T_NOMBRE = @T_NOMBRE,
        P_APELLIDO = @P_APELLIDO,
        S_APELLIDO = @S_APELLIDO,
        C_APELLIDO = @C_APELLIDO,
        EDAD = @EDAD,
        PRETENCION_SALARIO = @PRETENCION_SALARIO,
```

```
        ID_ESTADO_CIVIL = @ID_ESTADO_CIVIL,
        L_NACIMIENTO = @L_NACIMIENTO,
        F_NACIMIENTO = @F_NACIMIENTO,
        ID_GENERO = @ID_GENERO,
        ID_ETNIA = @ID_ETNIA,
        NACIONALIDAD = @NACIONALIDAD,
        ID_RELIGION = @ID_RELIGION,
        DPI = @DPI,
        ID_MUNICIPIO = @ID_MUNICIPIO,
        IGSS = @IGSS,
        NIT = @NIT,
        Foto = @Foto,
        LICENCIA = @LICENCIA,
        TIPO_LICENCIA = @TIPO_LICENCIA
    WHERE
        IdPersona = @IdPersona;
       END;
```

Procedimiento almacenado de Buscar RHPersona.

```
USE [encaagro]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[sp_BuscarPersona]
    @DPI VARCHAR(15)
AS
BEGIN
    SELECT
        P.IdPersona AS ID,
        P.P_NOMBRE,
        P.S_NOMBRE,
        P.T_NOMBRE,
        P.P_APELLIDO,
        P.S_APELLIDO,
        P.C_APELLIDO,
        P.EDAD,
        P.PRETENCION_SALARIO,
        T.ESTADO_CIVIL,
        P.L_NACIMIENTO,
        P.F_NACIMIENTO,
        G.GENERO,
        E.ETNIA,
        P.NACIONALIDAD,
        R.RELIGION,
        P.DPI,
        M.MUNICIPIO,
        P.IGSS,
        P.NIT,
        P.Foto,
        P.LICENCIA,
        P.TIPO_LICENCIA
    FROM
       RHPersona P
```

```
        INNER JOIN TYETNIA E ON P.ID_ETNIA = E.ID_ETNIA
        INNER JOIN TYRELIGION R ON P.ID_RELIGION = R.ID_RELIGION
        INNER JOIN TYGENERO G ON P.ID_GENERO = G.ID_GENERO
        INNER JOIN TYMUNICIPIO M ON P.ID_MUNICIPIO = M.ID_MUNICIPIO
        INNER JOIN TYESTADOCIVIL T ON P.ID_ESTADO_CIVIL = T.ID_ESTADO_CIVIL
    WHERE
         P.DPI LIKE @DPI + '%';
       END;
```

Procedimiento almacenado de listar RHLocalizacion.

```
USE [encaagro]
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[sp_MostrarRHLocalizacion]
AS
BEGIN
    SELECT R.Id_localizacion AS ID,
           P.DPI,
           T.Descripcion,
           R.Localizacion
    FROM RHLOCALIZACION R
    INNER JOIN RHPersona P ON P.IdPersona = R.IDPERSONA
    INNER JOIN TYLOCALIZACION T ON T.IDLOCALIZACION = R.ID_LOCALIZACION;
       END;
```

Procedimiento almacenado de insertar RHLocalizacion.

```
USE [encaagro]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[sp_InsertarRHLocalizacion]
    @IDPersona INT,
    @IDLocalizacion INT,
    @Localizacion VARCHAR(25)
AS
BEGIN
    INSERT INTO RHLOCALIZACION (IDPERSONA, IDLOCALIZACION, LOCALIZACION)
    VALUES (@IDPersona, @IDLocalizacion, @Localizacion);
       END;
```

Procedimiento almacenado de actualizar RHLocalizacion.

```
USE [encaagro]
GO
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[sp_ActualizarRhLocalizacion]
    @ID INT,
    @IDPersona INT,
    @IDLocalizacion INT,
    @Localizacion VARCHAR(25)
AS
BEGIN
    UPDATE RHLOCALIZACION
    SET IDPERSONA = @IDPersona,
        IDLOCALIZACION = @IDLocalizacion,
        LOCALIZACION = @Localizacion
    WHERE ID_LOCALIZACION = @ID;
       END;
```

Procedimiento almacenado de buscar RHLocalizacion

```
USE [encaagro]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[sp_BuscarRHLocalizacion]
    @DPI varchar(15)
AS
BEGIN
    SELECT R.Id_localizacion AS ID,
           P.DPI,
           T.Descripcion,
           R.Localizacion
    FROM RHLOCALIZACION R
    INNER JOIN RHPersona P ON P.IdPersona = R.IDPERSONA
    INNER JOIN TYLOCALIZACION T ON T.IDLOCALIZACION = R.ID_LOCALIZACION
    WHERE P.DPI  LIKE '%' + @DPI + '%';
       END;
```

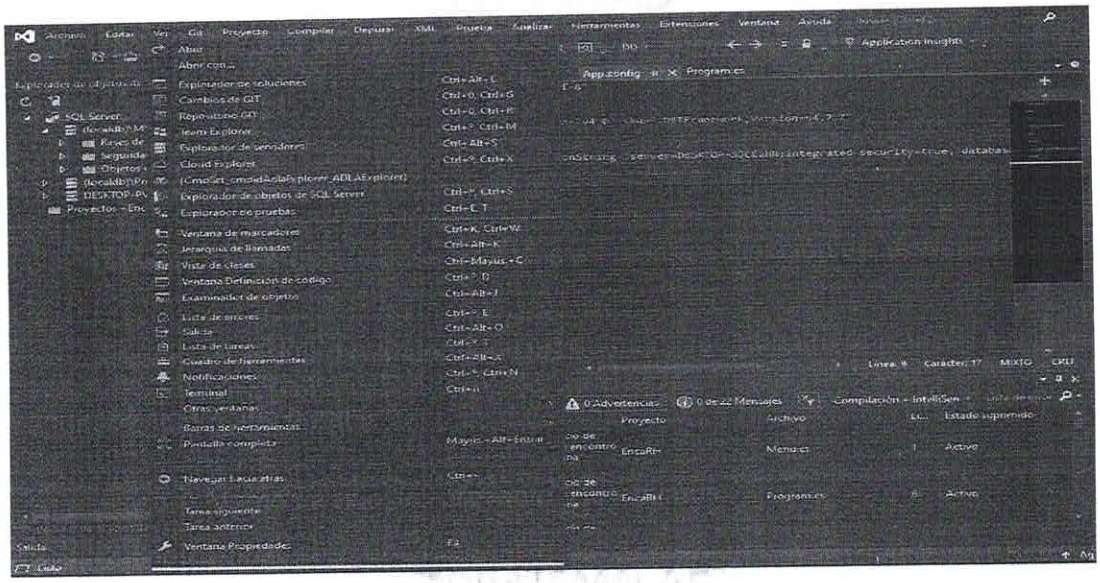
Con esta lógica se crean todos los procedimientos almacenados para cada tabla RH. Esto para tener una base de datos más robusta y que sea transaccional a la hora de tener nuestro servidor de base de datos levantado y funcionando que los usuarios que se conecten a nuestro servidor por medio de la aplicación puedan hacer sus transacciones de datos de una forma efectiva y no saturar nuestro servidor de base de datos.

Conexión de muestra base de datos a nuestra aplicación en Visual estudio.

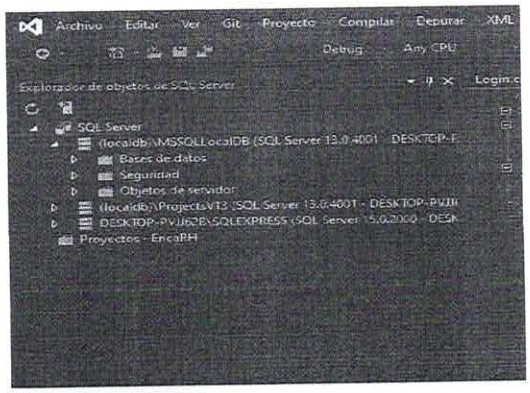
Para la conexión a nuestra base de datos desde nuestra aplicación se debe de seguir los siguientes pasos.

Paso 1. Vamos a nuestro proyecto y lo abrimos.

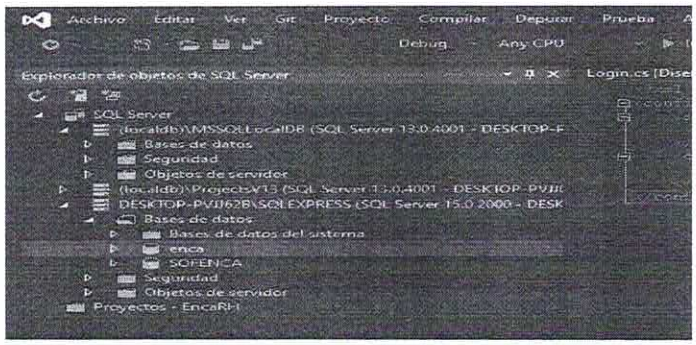
Nos vamos a la opción de VER en nuestro menú y seleccionamos EXPLORADOR DE OBJETOS DE SQL SERVER.



Al seleccionar esta opción nos muestra las conexiones que tenemos disponibles.



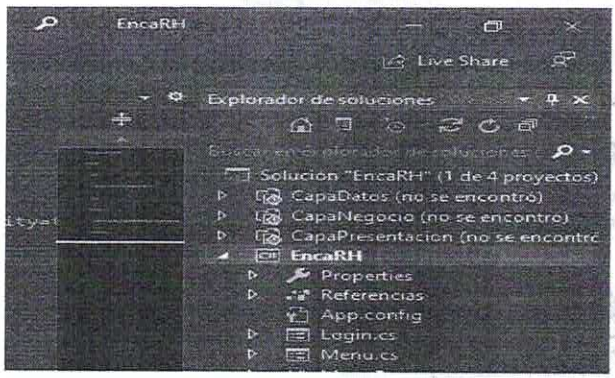
Seleccionamos BASES DE DATOS.



Y nos muestra la base datos que estamos trabajando. La seleccionamos y luego nos dirigimos a las propiedades y buscamos el string de conexión de la base de datos.

```
connectionString="server=DESKTOP-3JCE2HN;integrated security=true; database=ENCA"/>
```

Al tener nuestro string de conexion vamos a la carpeta de nuestro Proyecto.

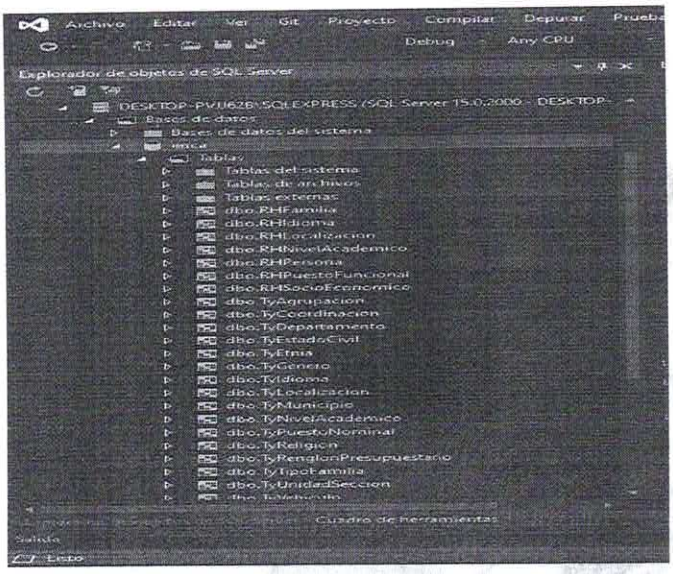


Nos dirigimos al archivo App.config.

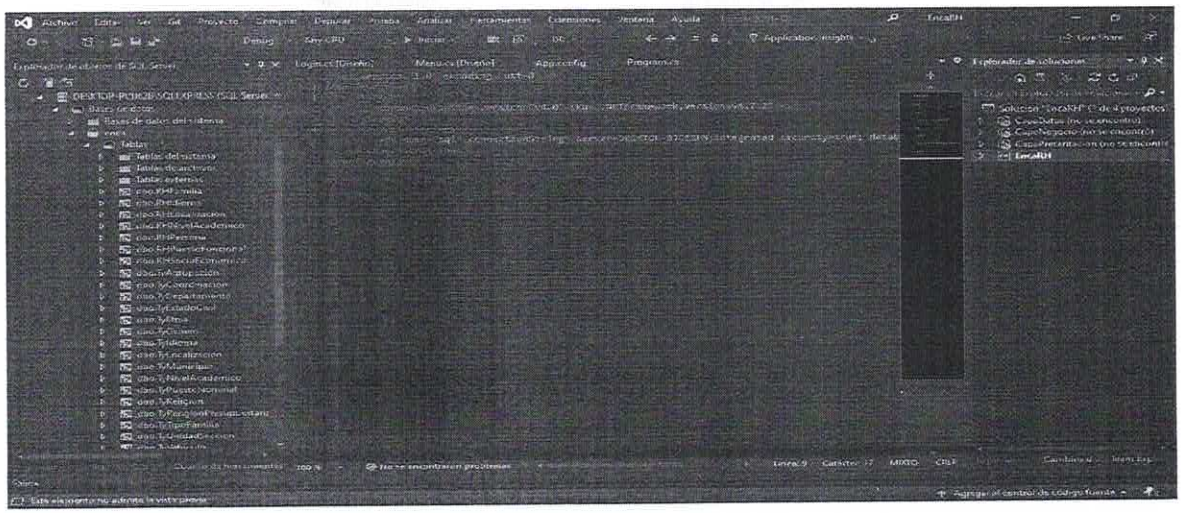
Y colocamos el siguiente código de conexión.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
    </startup>
       <connectionStrings>
             <add name="sql" connectionString="server=DESKTOP-3JCE2HN;integrated
security=true; database=ENCA"/>
       </connectionStrings>
</configuration>
```

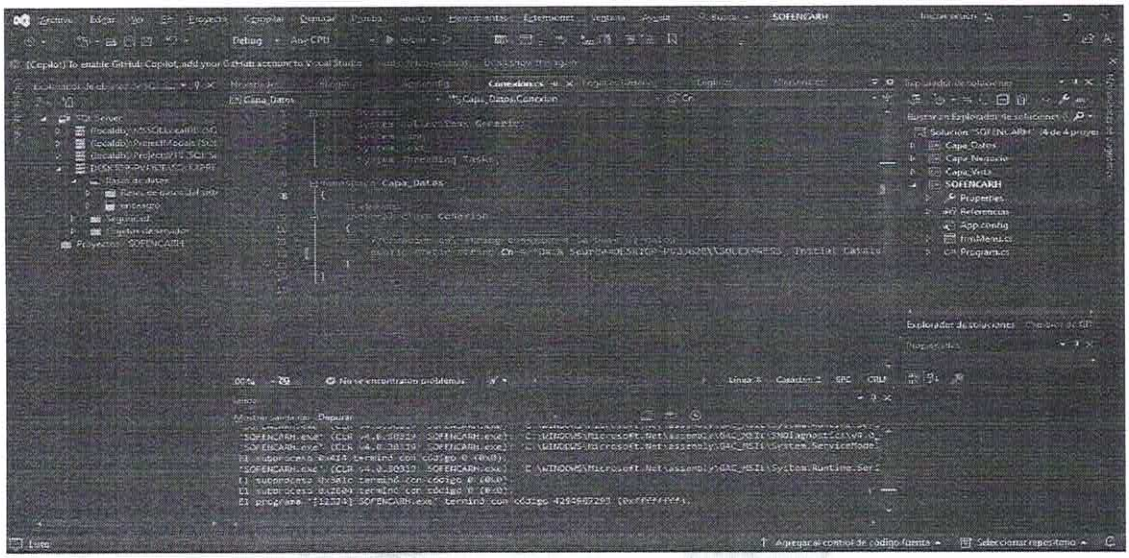
Con el string de conexión a nuestra base de datos y listo ya tenemos conexión a nuestra base de datos ya lista para desarrollar nuestra aplicación según la lógica de nuestra base de datos.



Ya tenemos nuestra base lista y sus tablas.

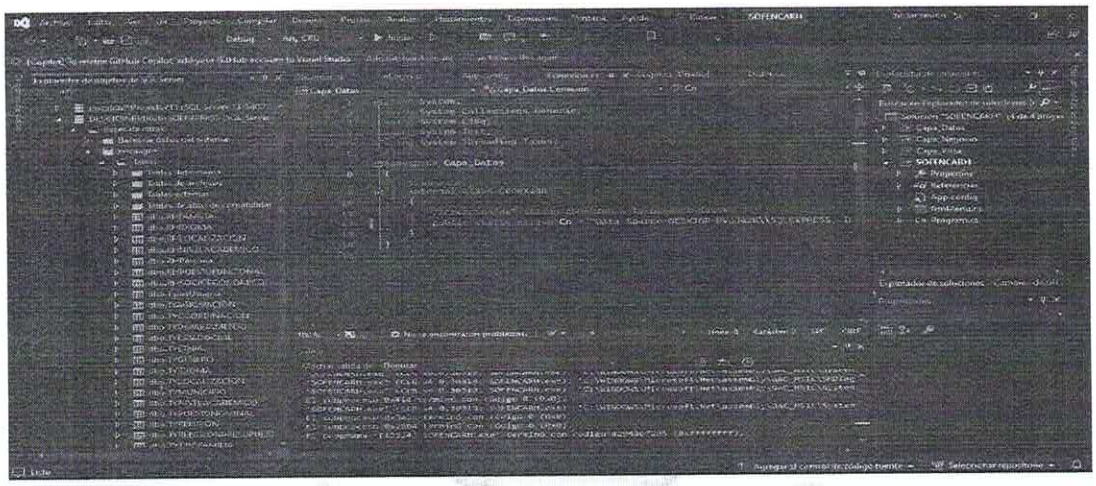


En mi caso voy hacer una conexión aparte esto para poder facilitar mi conexión a todas las tablas donde usare el string de conexión.



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

namespace Capa_Datos
{
    internal class Conexion
    {
        //Creacion del string conexion a la base de datos directo.
        public static string Cn = "Data Source=DESKTOP-PVJJ62B\\SQLEXPRESS; Initial
Catalog=encaagro; Integrated Security=true";
    }
}
```

Esto me ayudara a poder solo llamar a string de conexión con el nombre de Cn.



Creación de arquitectura de capas para el desarrollo de nuestra aplicación.

La arquitectura de capas es un enfoque de diseño de software que organiza el código en diferentes capas lógicas o niveles de abstracción, cada una con una responsabilidad específica. Este enfoque facilita el mantenimiento, la escalabilidad y la modularidad del código, ya que separa las preocupaciones y promueve la reutilización del código. Para una aplicación de Windows Forms en C#, las capas típicas pueden incluir:

1. Capa de Presentación (UI):

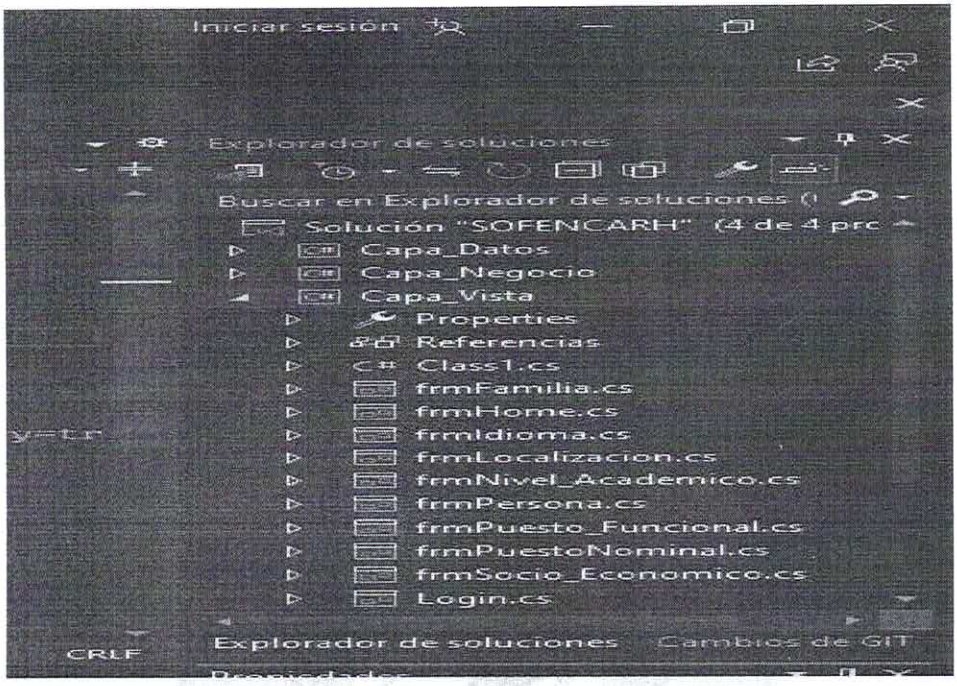
Descripción: Esta capa maneja la interfaz de usuario y la interacción con el usuario final. En el caso de Windows Forms, esta capa consiste en formularios y controles que muestran la información y reciben la entrada del usuario.

Métodos:

Manejo de eventos del usuario (clics de botón, cambios de selección, etc.).

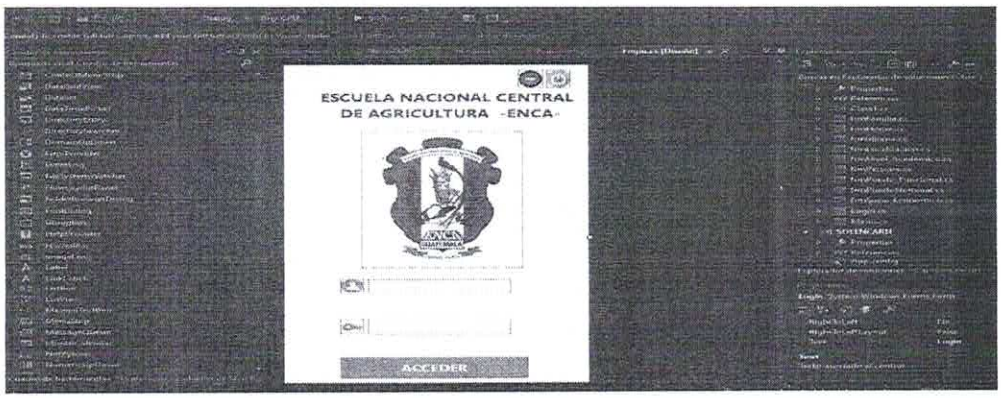
Actualización de la interfaz de usuario con datos de la capa de lógica.

Validación de entrada del usuario.

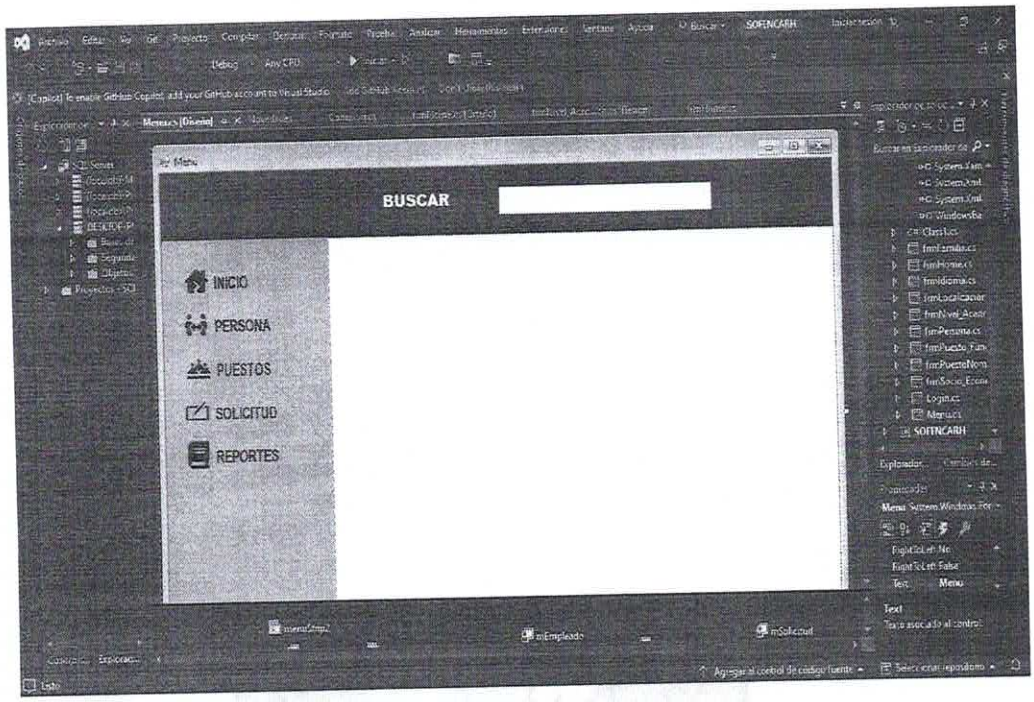


Están son las vistas del usuario aquí se tiene un diseño inicial con forme se valla creando la aplicación se va ir modificando los diseños de usuario.

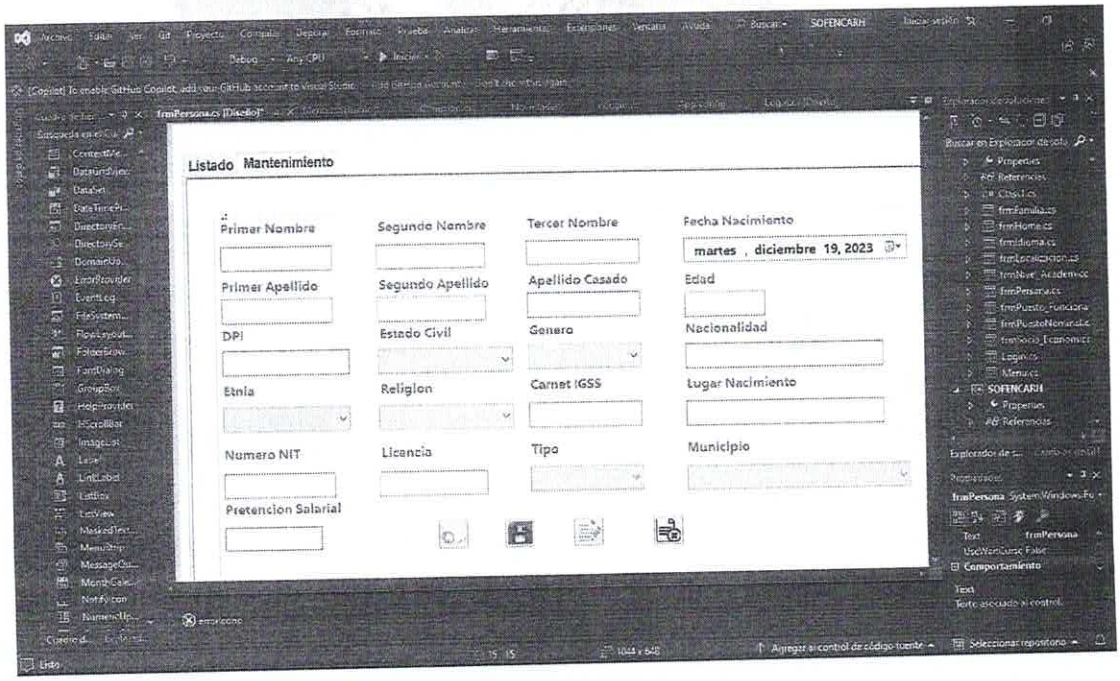
Login.



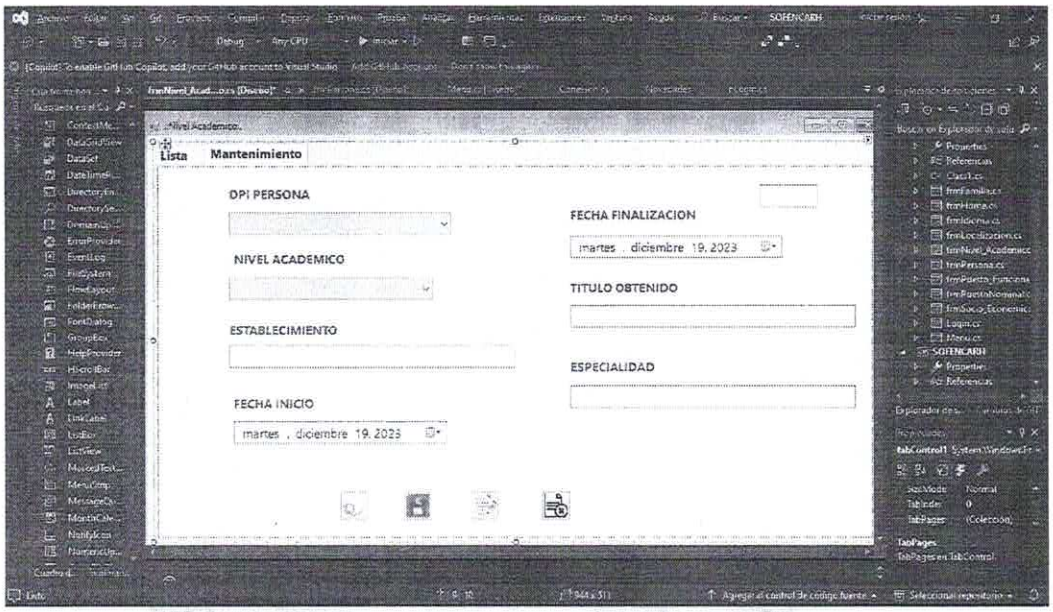
Menú.



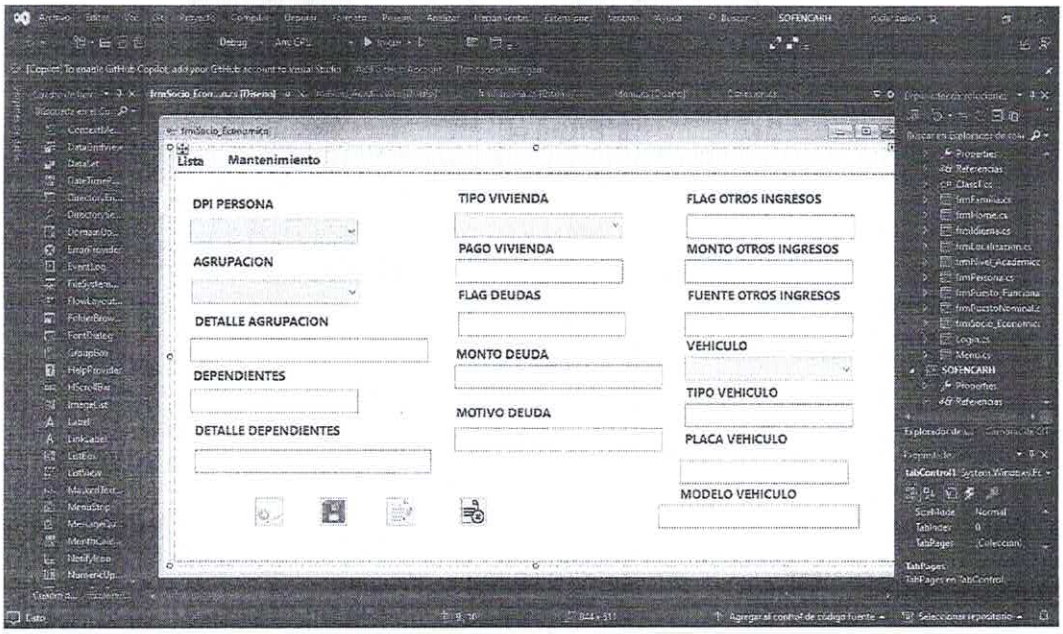
Frmpersona.



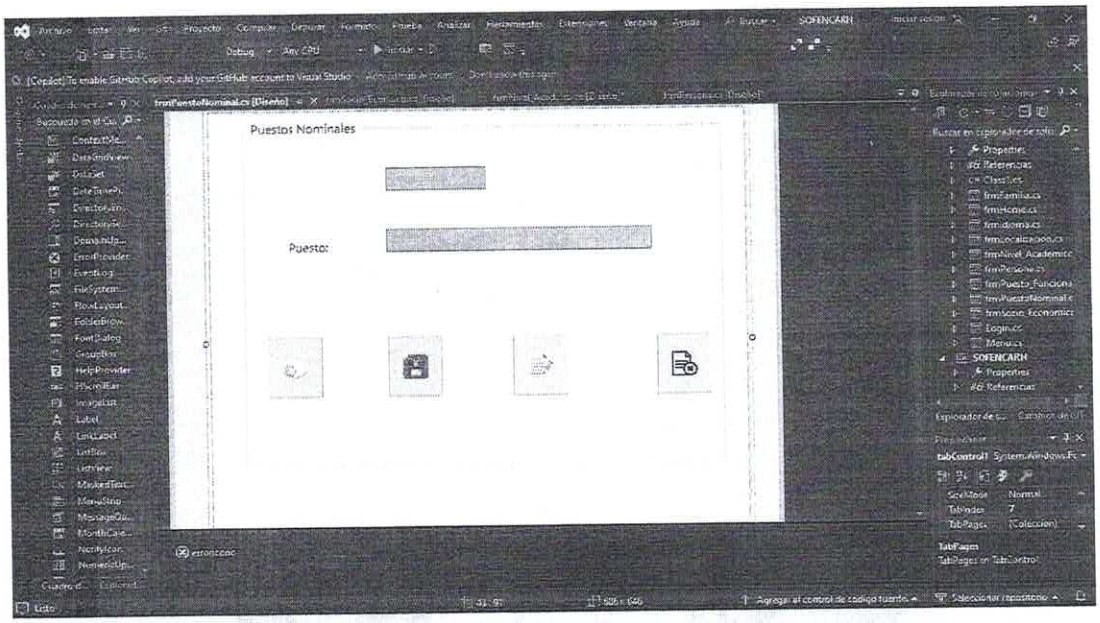
FrmNivelAcademico.



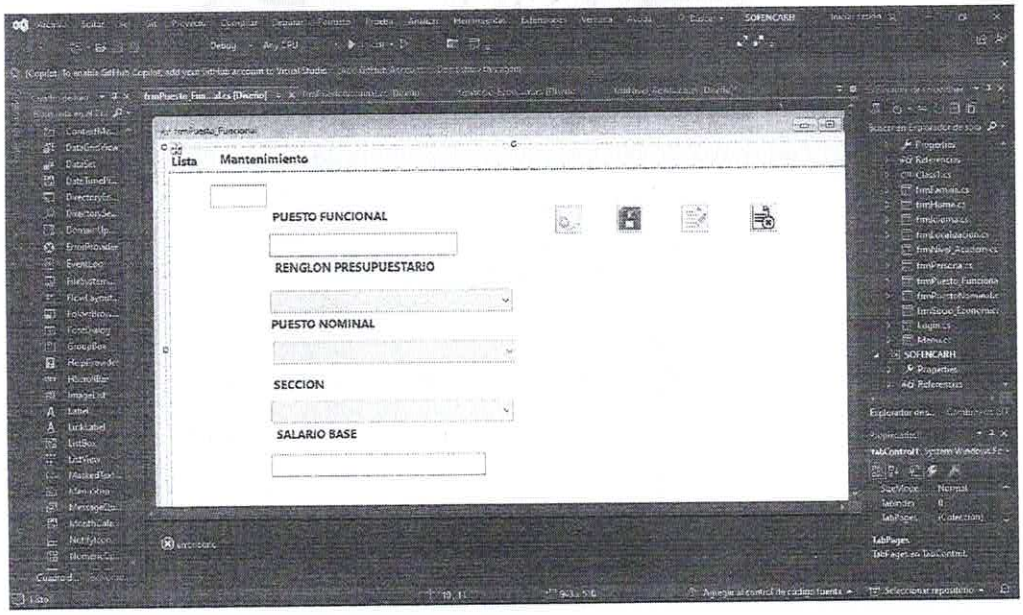
FrmSocioEconomico



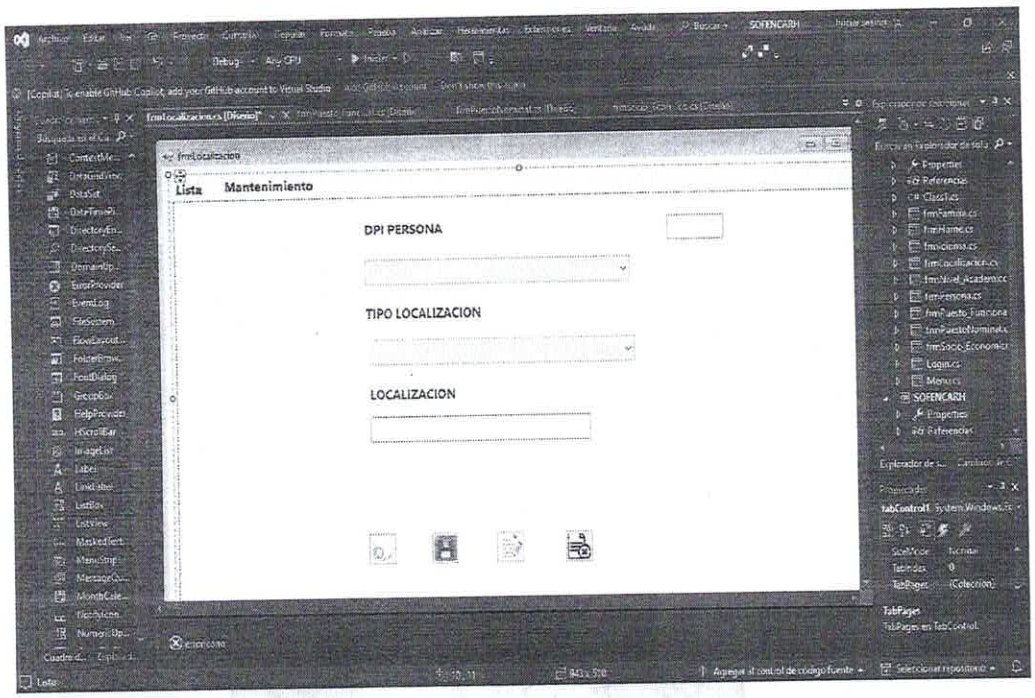
FrmpuestosNominal



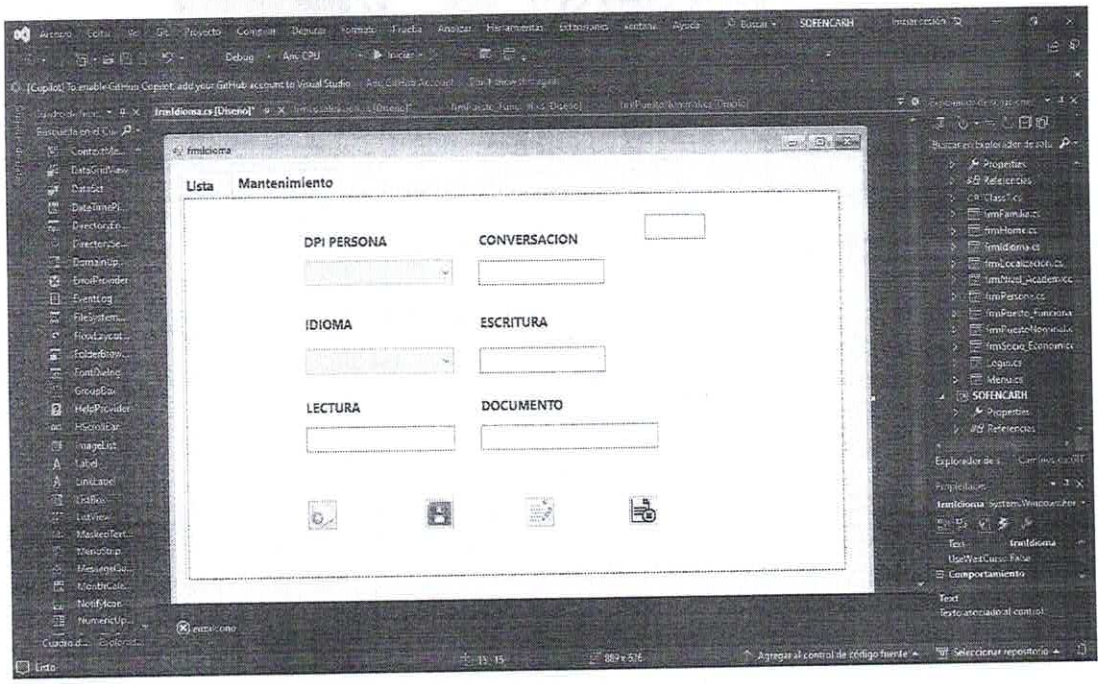
FrmpuestosFuncional



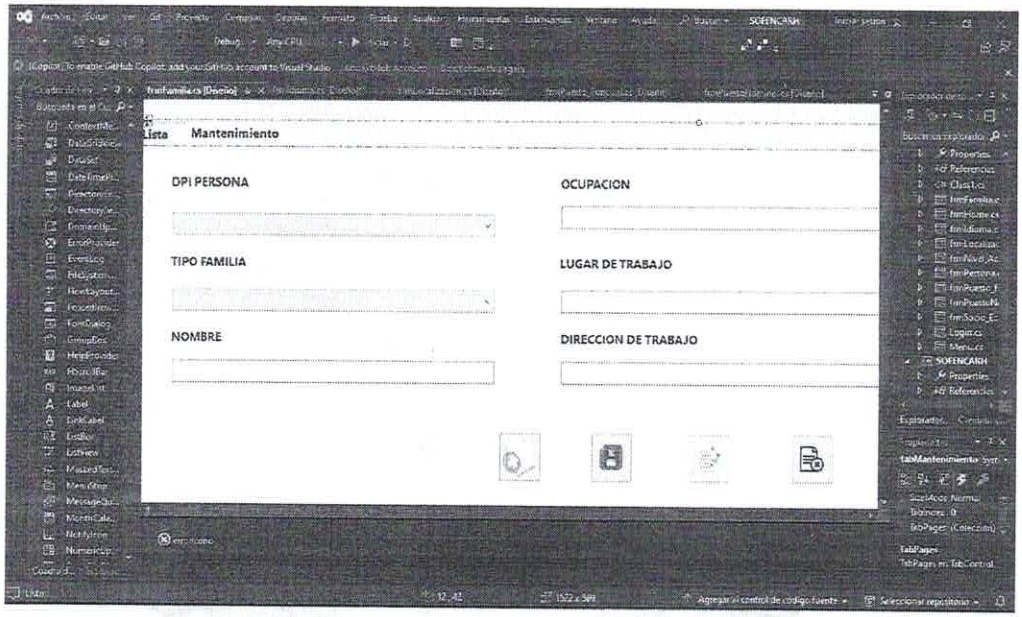
FrmLocalizacion.



FrmIdioma.



FrmFamilia.



Como podemos ver esta capa va centrada al usuario aquí se da el diseño a las pantallas de nuestra aplicación.

2. Capa de Lógica de Negocio (BLL - Business Logic Layer):

Descripción: Aquí se encuentra la lógica de negocio de la aplicación. Esta capa procesa y manipula los datos antes de enviarlos a la capa de datos o presentarlos en la interfaz de usuario.

Métodos:

Implementación de reglas de negocio.

Coordinación de operaciones entre diferentes entidades de datos.

Transformación de datos según sea necesario.

En esta capa se hace un modelo lógico entre la capa de vistas de usuario de la capa de datos. Para ello se crea un archivo llamado igual a su archivo de capa de vistas, pero este es .cs.

Login.cs

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

namespace Capa_Negocio
{
    public class nLogin
    {
        private cLogin  userRepository;

        public nLogin()
        {
            userRepository = new cLogin();
        }

        public bool AuthenticateUser(string username, string password, out bool
isValid, out string userType)
        {
            return userRepository.AuthenticateUser(username, password, out isValid,
out userType);
        }
    }
      }
```

Este hace referencia a los datos de la capa de datos.

Persona.cs

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

namespace Capa_Negocio
{
    public class nPersona
    {
        public static string InsertarPersona(
```

```
string P_NOMBRE, string S_NOMBRE, string T_NOMBRE,
string P_APELLIDO, string S_APELLIDO, string C_APELLIDO,
string EDAD, string PRETENCION_SALARIO, int ID_ESTADO_CIVIL,
string L_NACIMIENTO, DateTime F_NACIMIENTO, int ID_GENERO,
int ID_ETNIA, string NACIONALIDAD, int ID_RELIGION,
string DPI, int ID_MUNICIPIO, string IGSS, string NIT,
byte[] Foto, string LICENCIA, string TIPO_LICENCIA)
    {
        cPersona persona = new cPersona
        {
            P_NOMBRE = P_NOMBRE,
            S_NOMBRE = S_NOMBRE,
            T_NOMBRE = T_NOMBRE,
            P_APELLIDO = P_APELLIDO,
            S_APELLIDO = S_APELLIDO,
            C_APELLIDO = C_APELLIDO,
            EDAD = EDAD,
            PRETENCION_SALARIO = PRETENCION_SALARIO,
            ID_ESTADO_CIVIL = ID_ESTADO_CIVIL,
            L_NACIMIENTO = L_NACIMIENTO,
            F_NACIMIENTO = F_NACIMIENTO,
            ID_GENERO = ID_GENERO,
            ID_ETNIA = ID_ETNIA,
            NACIONALIDAD = NACIONALIDAD,
            ID_RELIGION = ID_RELIGION,
            DPI = DPI,
            ID_MUNICIPIO = ID_MUNICIPIO,
            IGSS = IGSS,
            NIT = NIT,
            Foto = Foto,
            LICENCIA = LICENCIA,
            TIPO_LICENCIA = TIPO_LICENCIA
        };

        return persona.Insertar(persona);
    }


    public static DataTable Buscar(string textoBuscar)
    {

        cPersona obj = new cPersona();
        obj.textoBuscar = textoBuscar;
        return obj.Buscar(obj);
    }

    public static DataTable MostrarPersonas()
    {
        return new cPersona().Mostrar();
    }

    public static string EditarPersona(
int idPersona,
string p_NOMBRE,
string s_NOMBRE,
string t_NOMBRE,
string p_APELLIDO,
```

```
    string s_APELLIDO,
    string c_APELLIDO,
    string eDAD,
    string pRETENCION_SALARIO,
    int iD_ESTADO_CIVIL,
    string l_NACIMIENTO,
    DateTime f_NACIMIENTO,
    int iD_GENERO,
    int iD_ETNIA,
    string nACIONALIDAD,
    int iD_RELIGION,
    string dPI,
    int iD_MUNICIPIO,
    string iGSS,
    string nIT,
    byte[] foto,
    string lICENCIA,
    string tIPO_LICENCIA)
        {
            cPersona Obj = new cPersona();
            Obj.IdPersona = idPersona;
            Obj.P_NOMBRE = p_NOMBRE;
            Obj.S_NOMBRE = s_NOMBRE;
            Obj.T_NOMBRE = t_NOMBRE;
            Obj.P_APELLIDO = p_APELLIDO;
            Obj.S_APELLIDO = s_APELLIDO;
            Obj.C_APELLIDO = c_APELLIDO;
            Obj.EDAD = eDAD;
            Obj.PRETENCION_SALARIO = pRETENCION_SALARIO;
            Obj.ID_ESTADO_CIVIL = iD_ESTADO_CIVIL;
            Obj.L_NACIMIENTO = l_NACIMIENTO;
            Obj.F_NACIMIENTO = f_NACIMIENTO;
            Obj.ID_GENERO = iD_GENERO;
            Obj.ID_ETNIA = iD_ETNIA;
            Obj.NACIONALIDAD = nACIONALIDAD;
            Obj.ID_RELIGION = iD_RELIGION;
            Obj.DPI = dPI;
            Obj.ID_MUNICIPIO = iD_MUNICIPIO;
            Obj.IGSS = iGSS;
            Obj.NIT = nIT;
            Obj.Foto = foto;
            Obj.LICENCIA = lICENCIA;
            Obj.TIPO_LICENCIA = tIPO_LICENCIA;

            return Obj.Actualizar(Obj);
        }


    }
    }
```

NivelAcademico.cs

```
using Capa_Datos;
using System;
using System.Collections.Generic;
using System.Data;
```

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

namespace Capa_Negocio
{
    public  class nNivelAcademico
    {

        public static DataTable Mostrar()
        {
            return new cNivelAcademico().Mostrar();
        }

        public static DataTable Buscar(string textoBuscar)
        {
            cNivelAcademico obj = new cNivelAcademico();
            obj.TxtBuscar = textoBuscar;
            return obj.Buscar(obj);
        }

        public static string Insertar( int idpersona, int idAcademico,
            string establecimiento, DateTime dFecha, DateTime aFecha,
            string titulo, string especialidad)
        {
            cNivelAcademico Obj = new cNivelAcademico();
            Obj.Id_Persona = idpersona;
            Obj.Id_Academico = idAcademico;
            Obj.Establecimiento = establecimiento;
            Obj.Fecha_Inicio = dFecha;
            Obj.Fecha_Outicio = aFecha;
            Obj.Titulo = titulo;
            Obj.Especialidad = especialidad;

            return Obj.Insertar(Obj);
        }

        public static string Actualizar(int idNivel,int idpersona, int idAcademico,
          string establecimiento, DateTime dFecha, DateTime aFecha,
          string titulo, string especialidad)
        {
            cNivelAcademico Obj = new cNivelAcademico();
            Obj.Id_Nivel = idNivel;
            Obj.Id_Persona = idpersona;
            Obj.Id_Academico = idAcademico;
            Obj.Establecimiento = establecimiento;
            Obj.Fecha_Inicio = dFecha;
            Obj.Fecha_Outicio = aFecha;
            Obj.Titulo = titulo;
            Obj.Especialidad = especialidad;

            return Obj.Actualizar(Obj);
        }

    }
    }
```

PuestoFuncional.cs

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

namespace Capa_Negocio
{
    public class nPuestoFuncional
    {
        public static string InsertarPuestoFuncional(
        string PuestoFuncional, int IdRenglon, int IdPuestoNominal, int
IdUnidadSeccion, string SalarioBase)
        {
            cPuestoFuncional puestoFuncional = new cPuestoFuncional
            {
                PuestoFuncional = PuestoFuncional,
                IdRenglon = IdRenglon,
                IdPuestoNominal = IdPuestoNominal,
                IdUnidadSeccion = IdUnidadSeccion,
                SalarioBase = SalarioBase
            };

            return puestoFuncional.Insertar(puestoFuncional);
        }

        public static DataTable Buscar(string textoBuscar)
        {
            cPuestoFuncional obj = new cPuestoFuncional();
            obj.Puesto = textoBuscar;
            return obj.Buscar(obj);
        }

        public static DataTable MostrarPuestosFuncionales()
        {
            return new cPuestoFuncional().Mostrar();
        }

        public static string EditarPuestoFuncional(
            int idPuestoFuncional, string PuestoFuncional, int IdRenglon, int
IdPuestoNominal, int IdUnidadSeccion, string SalarioBase)
        {
            cPuestoFuncional Obj = new cPuestoFuncional();
            Obj.IdPuestoFuncional = idPuestoFuncional;
            Obj.PuestoFuncional = PuestoFuncional;
            Obj.IdRenglon = IdRenglon;
            Obj.IdPuestoNominal = IdPuestoNominal;
            Obj.IdUnidadSeccion = IdUnidadSeccion;
            Obj.SalarioBase = SalarioBase;

            return Obj.Actualizar(Obj);
        }
    }
```

```
    }
```

PuestosNominales.cs

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

namespace Capa_Negocio
{
    public class nPuestoNominal
    {

        public static string Insertar(string puestoNominal)
        {
            cPuestoNominal Obj = new cPuestoNominal();
            Obj.PuestoNominal = puestoNominal;

            return Obj.Insertar(Obj);
        }

        public static string Editar(int idPuestoNominal, string puestoNominal)
        {
            cPuestoNominal Obj = new cPuestoNominal();
            Obj.IdPuestoNominal = idPuestoNominal;
            Obj.PuestoNominal = puestoNominal;


            return Obj.Editar(Obj);
        }

        public static string Eliminar(int idPuestoNominal)
        {
            cPuestoNominal Obj = new cPuestoNominal();
            Obj.IdPuestoNominal = idPuestoNominal;
            return Obj.Eliminar(Obj);
        }

        public static DataTable Mostrar()
        {
            return new cPuestoNominal().Mostrar();
        }

        public static DataTable Buscar(string textoBuscar)
        {
            cPuestoNominal obj = new cPuestoNominal();
            obj.TextoBuscar = textoBuscar;
            return obj.BuscarPuestoNominal(obj);
        }
    }
    }
```

Localizacion.cs

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

namespace Capa_Negocio
{
    public class nLocalizacion
    {
        public static string InsertarLocalizacion(
        int idPersona, int idLocalizacionTipo, string localizacion)
        {
            cLocalizacion localizacionDatos = new cLocalizacion
            {
                IdPersona = idPersona,
                IdLocalizacionTipo = idLocalizacionTipo,
                Localizacion = localizacion,
            };

            return localizacionDatos.Insertar(localizacionDatos);
        }

        public static DataTable BuscarLocalizacion(string dpi)
        {
            cLocalizacion obj = new cLocalizacion();
            obj.DPI = dpi;
            return obj.Buscar(obj);
        }

        public static DataTable MostrarLocalizaciones()
        {
            return new cLocalizacion().Mostrar();
        }

        public static string ActualizarLocalizacion(
            int idLocalizacion, int idPersona, int idLocalizacionTipo,
            string localizacion)
        {
            cLocalizacion obj = new cLocalizacion();
            obj.IdLocalizacion = idLocalizacion;
            obj.IdPersona = idPersona;
            obj.IdLocalizacionTipo = idLocalizacionTipo;
            obj.Localizacion = localizacion;

            return obj.Actualizar(obj);
        }
    }
    }
```

Idioma.cs

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

namespace Capa_Negocio
{
    public class nIdioma
    {

        public static string Insertar
            (int idPersona,int ididioma,string conversacion,string escritura, string
lectura, string documento)
        {
            cIdioma Idioma = new cIdioma
            {
                IdPersona = idPersona,
                IdIdioma = ididioma,
                Conversacion = conversacion,
                Escritura = escritura,
                Lectura = lectura,
                Documento = documento

            };

            return Idioma.Insertar(Idioma);
        }


        public static DataTable MostarrhIdioma()
        {
            return new cIdioma().Mostrar();
        }

        public static DataTable Buscar(string txtDpi)
        {
            cIdioma obj = new cIdioma();
            obj.TxtDpi = txtDpi;
            return obj.Buscar(obj);
        }

        public static string Actualizar
           (int id_Idioma,int idPersona, int ididioma, string conversacion, string
escritura, string lectura, string documento)
        {
            cIdioma Idioma = new cIdioma
            {
                Id_Idioma = id_Idioma,
                IdPersona = idPersona,
                IdIdioma = ididioma,
                Conversacion = conversacion,
                Escritura = escritura,
                Lectura = lectura,
                Documento = documento
```

```
            };

            return Idioma.Actualizar(Idioma);
        }

    }
      }
```

Familia.cs

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

namespace Capa_Negocio
{
    public class nFamilia
    {

        public static string EditarFamilia(
    int id,
    int idPersona,
    int idTipoFamilia,
    string nombre,
    DateTime fNacimiento,
    string ocupacion,
    string telefono,
    string lTrabajo,
    string dTrabajo,
    string tTrabajo)
        {
            // Crear una instancia de la clase Familia (suponiendo que ya tienes
una)
            cFamilia obj = new cFamilia();

            // Establecer las propiedades del objeto con los valores proporcionados
            obj.Id_Familia = id;
            obj.IdPersona = idPersona;
            obj.IdTipo_Familia = idTipoFamilia;
            obj.Nombre = nombre;
            obj.F_nacimiento = fNacimiento;
            obj.Ocupacion = ocupacion;
            obj.Telefono = telefono;
            obj.L_trabajo = lTrabajo;
            obj.D_trabajo = dTrabajo;
            obj.T_trabajo = tTrabajo;

            // Llamar al método de actualización en la clase Familia (debes tener
este método)
            return obj.Actualizar(obj);
        }

        public static DataTable Buscar(string textoBuscar)
```

```
        {

            cFamilia obj = new cFamilia();
            obj.DPI = textoBuscar;
            return obj.Buscar(obj);
        }

        public static DataTable MostrarPersonas()
        {
            return new cFamilia().Mostrar();
        }


        public static string Insertar(
    int idPersona,
    int idTipoFamilia,
    string nombre,
    DateTime fNacimiento,
    string ocupacion,
    string telefono,
    string lTrabajo,
    string dTrabajo,
    string tTrabajo)
        {
            // Crear una instancia de la clase Familia (suponiendo que ya tienes
una)
            cFamilia obj = new cFamilia();

            // Establecer las propiedades del objeto con los valores proporcionados

            obj.IdPersona = idPersona;
            obj.IdTipo_Familia = idTipoFamilia;
            obj.Nombre = nombre;
            obj.F_nacimiento = fNacimiento;
            obj.Ocupacion = ocupacion;
            obj.Telefono = telefono;
            obj.L_trabajo = lTrabajo;
            obj.D_trabajo = dTrabajo;
            obj.T_trabajo = tTrabajo;

            // Llamar al método de actualización en la clase Familia (debes tener
este método)
            return obj.Insertar(obj);
        }
    }


    }
```

En términos generales esta capa hace una referencia a la vista a los métodos de insertar, mostrar actualizar y buscar los datos de cada pantalla y a su vez esta va enlazada a la capa más importante de la arquitectura de capas que es la de capa de datos que a continuación la veremos.

3. Capa de Acceso a Datos (DAL - Data Access Layer):

Descripción: La capa de acceso a datos se encarga de interactuar con la fuente de datos, que es nuestra base de datos que creamos y diseñamos nuestras relaciones entre las tablas.

Métodos:

Consultas y actualizaciones de la base de datos.

Mapeo objeto-relacional (ORM) para traducir datos entre el modelo de datos y las estructuras de objetos en el código.

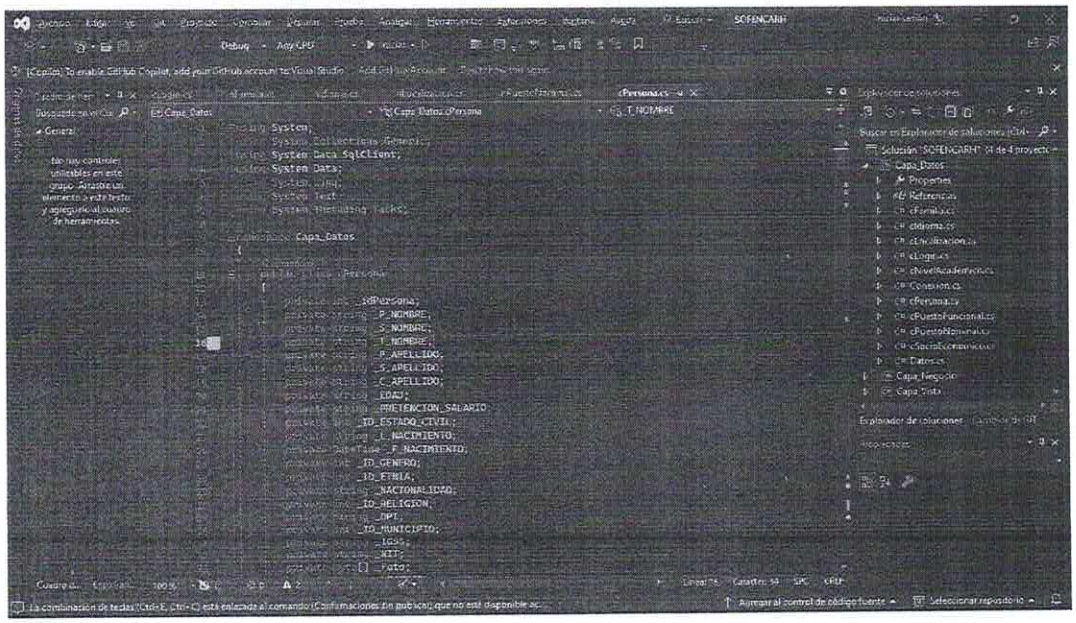
En esta capa usaremos nuestro string de conexión que se creó anterior mente porque aquí se interactúa directamente con nuestra base de datos. El código que se crea aquí es un poco largo ya que se realizan todos los métodos del CRUD de una base de datos.

Login.cs (solo se usa un método de validación).

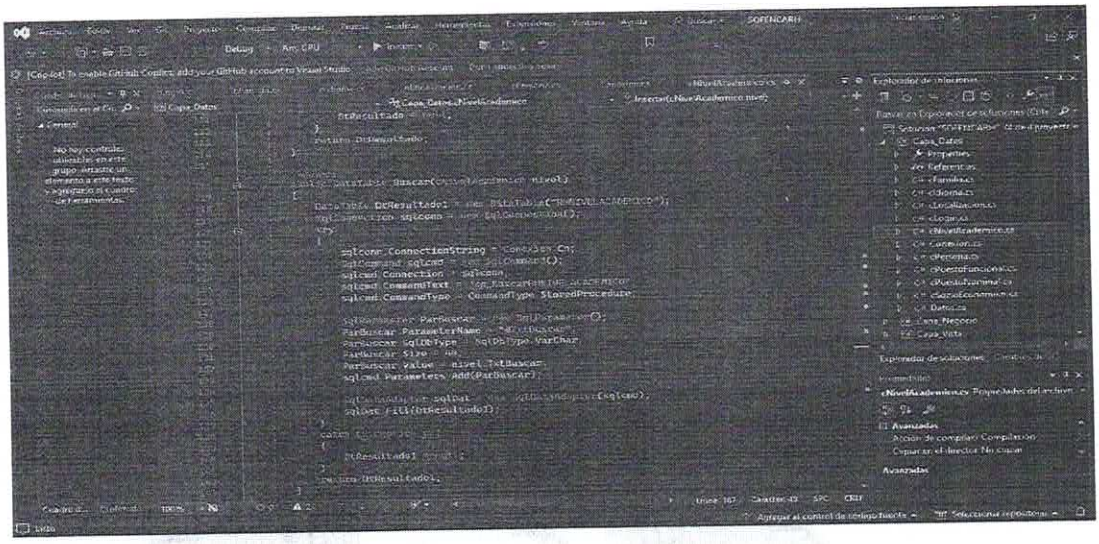
```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Capa_Datos
{
    public class cLogin
    {
        public bool AuthenticateUser(string username, string password, out bool
isValid, out string userType)
        {
            using (SqlConnection connection = new SqlConnection(Conexion.Cn))
            {
                connection.Open();

                using (SqlCommand command = new SqlCommand("sp_AutenticarUsuario",
connection))
```

```
                {
                    command.CommandType = CommandType.StoredProcedure;

                    command.Parameters.AddWithValue("@NombreUsuario", username);
                    command.Parameters.AddWithValue("@Contrasena", password);

                    SqlParameter isValidParameter = new SqlParameter("@EsValido",
SqlDbType.Bit);
                    isValidParameter.Direction = ParameterDirection.Output;
                    command.Parameters.Add(isValidParameter);

                    SqlParameter userTypeParameter = new
SqlParameter("@TipoUsuario", SqlDbType.NVarChar, 50);
                    userTypeParameter.Direction = ParameterDirection.Output;
                    command.Parameters.Add(userTypeParameter);

                    command.ExecuteNonQuery();

                    isValid = (bool)isValidParameter.Value;
                    userType = userTypeParameter.Value.ToString();

                    return isValid;
                }
            }
        }

    }
}
```

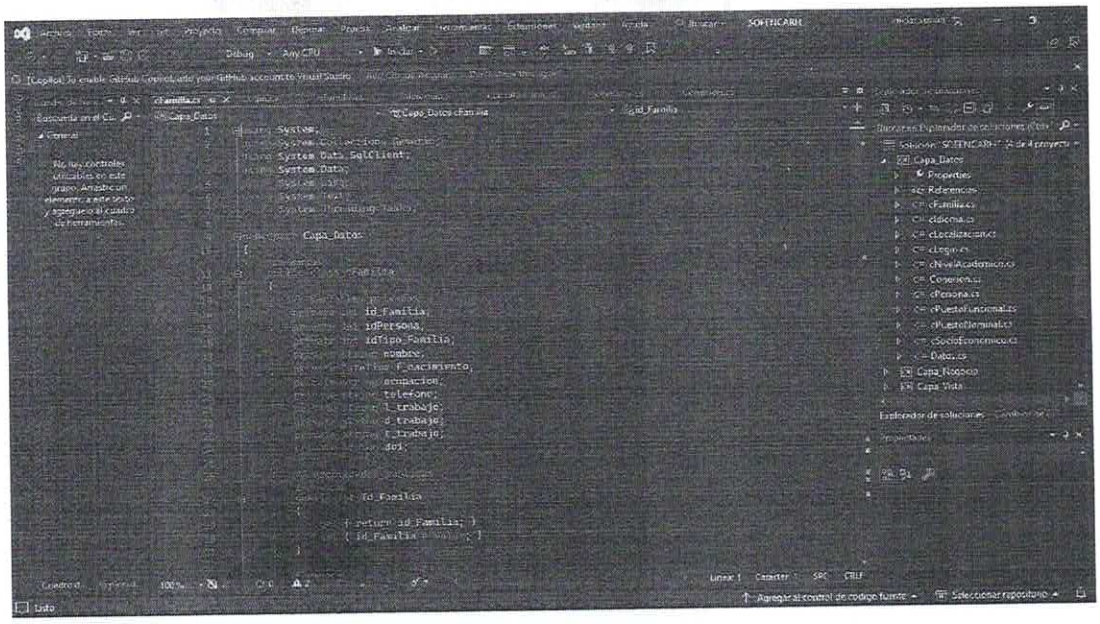
Persona.cs (métodos de CRUD de base de datos).



NivelAcadenico.

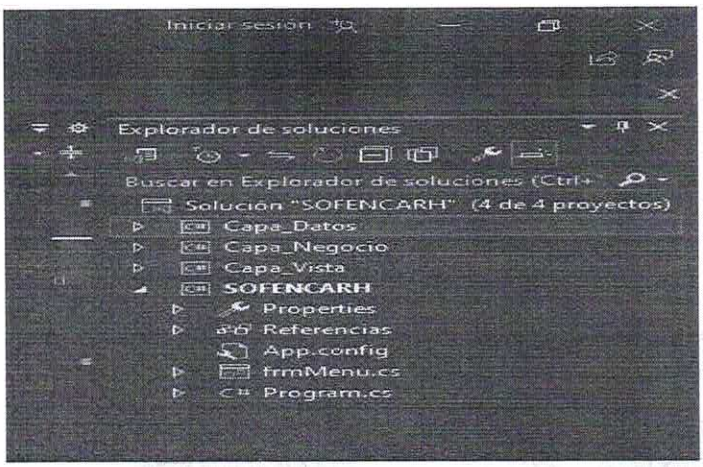


Familia.cs



No coloque el código de todas las pantallas porque es demasiado las líneas de código que tiene cada fichero para cada método del CRUD de la base de datos.

En términos generales la aplicación cuenta de arquitectura de capas que es segura confiable robusta y eficiente para su ejecución.



La ventaja de esta arquitectura es que cada capa tiene una responsabilidad clara y se puede modificar sin afectar directamente a las demás capas, siempre y cuando se respeten las interfaces definidas entre ellas. Además, facilita la prueba unitaria, ya que cada capa se puede probar de forma independiente.

Lógica de programación de funcionamiento del CRUD de cada pantalla, diseño de pantallas y ejecución de avance de la aplicación.

Con la lógica de programación de capas se hará un breve resumen de como es el funcionamiento de este tipo de arquitectura.

Cuando trabajas con Windows Forms y C# en una arquitectura de capas, la idea principal es separar la lógica de presentación (UI) de la lógica de negocio y acceso a datos. A continuación, te proporcionaré un ejemplo básico de cómo estructurar el

CRUD en una aplicación de Windows Forms con una arquitectura de capas en C# que se adaptara a nuestra aplicación.

1. Capa de Datos:

Clase de Acceso a Datos (Data Access Layer - DAL):

Codigo.

```
public class DataAccessLayer
{
   public List<Elemento> ObtenerTodosLosElementos()
   {
      // Lógica para obtener todos los elementos de la base de datos
   }

   public Elemento ObtenerElementoPorId(int id)
   {
      // Lógica para obtener un elemento por su ID de la base de datos
   }

   public void InsertarElemento(Elemento nuevoElemento)
   {
      // Lógica para insertar un nuevo elemento en la base de datos
   }

   public void ActualizarElemento(Elemento elementoActualizado)
   {
      // Lógica para actualizar un elemento en la base de datos
   }
```

```
    public void EliminarElemento(int id)
    {
        // Lógica para eliminar un elemento de la base de datos
    }
}
```

2. Capa de Negocios (Business Logic Layer - BLL):

Clase de Lógica de Negocios:

```
public class BusinessLogicLayer
{
    private DataAccessLayer _dataAccessLayer;

    public BusinessLogicLayer()
    {
        _dataAccessLayer = new DataAccessLayer();
    }

    public List<Elemento> ObtenerTodosLosElementos()
    {
        return _dataAccessLayer.ObtenerTodosLosElementos();
    }

    public Elemento ObtenerElementoPorId(int id)
    {
        return _dataAccessLayer.ObtenerElementoPorId(id);
    }
```

```
public void InsertarElemento(Elemento nuevoElemento)
{
    // Lógica de validación si es necesario
    _dataAccessLayer.InsertarElemento(nuevoElemento);
}

public void ActualizarElemento(Elemento elementoActualizado)
{
    // Lógica de validación si es necesario
    _dataAccessLayer.ActualizarElemento(elementoActualizado);
}

public void EliminarElemento(int id)
{
    _dataAccessLayer.EliminarElemento(id);
}
}
```

3. Capa de Presentación (Windows Forms):

Formulario de Lista:

```
public partial class ListaForm : Form
{
    private BusinessLogicLayer _bll;
    public ListaForm()
    {
```

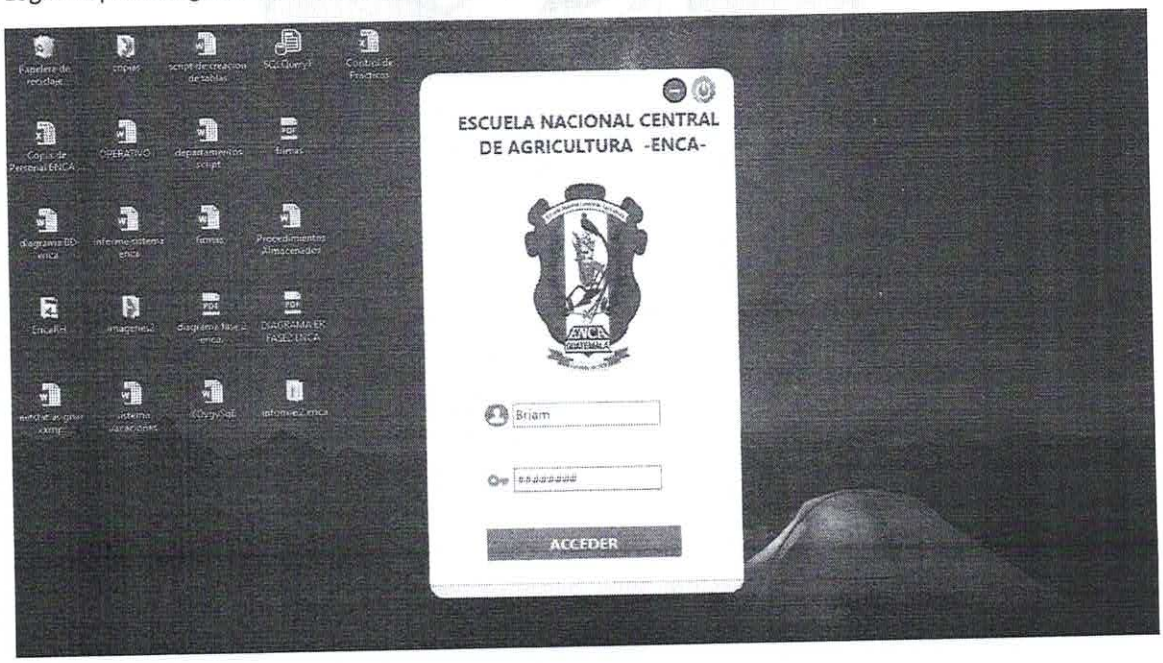
```
    InitializeComponent();
    _bll = new BusinessLogicLayer();
    CargarLista();
}

private void CargarLista()
{
    // Lógica para cargar la lista en el DataGridView

private void NuevoButton_Click(object sender, EventArgs e)
{
    // Mostrar el formulario de creación
}
private void EditarButton_Click(object sender, EventArgs e)
{
    // Obtener el ID seleccionado y mostrar el formulario de edición
}
private void EliminarButton_Click(object sender, EventArgs e)
```

```
    {
        // Obtener el ID seleccionado y eliminar el elemento
        CargarLista(); // Recargar la lista después de la eliminación
    }
}
```

Con esta lógica se hacen los botones que están en los formularios de cada pantalla para ejecutar los métodos de insertar, seleccionar, Actualizar, etc. De las pantallas.

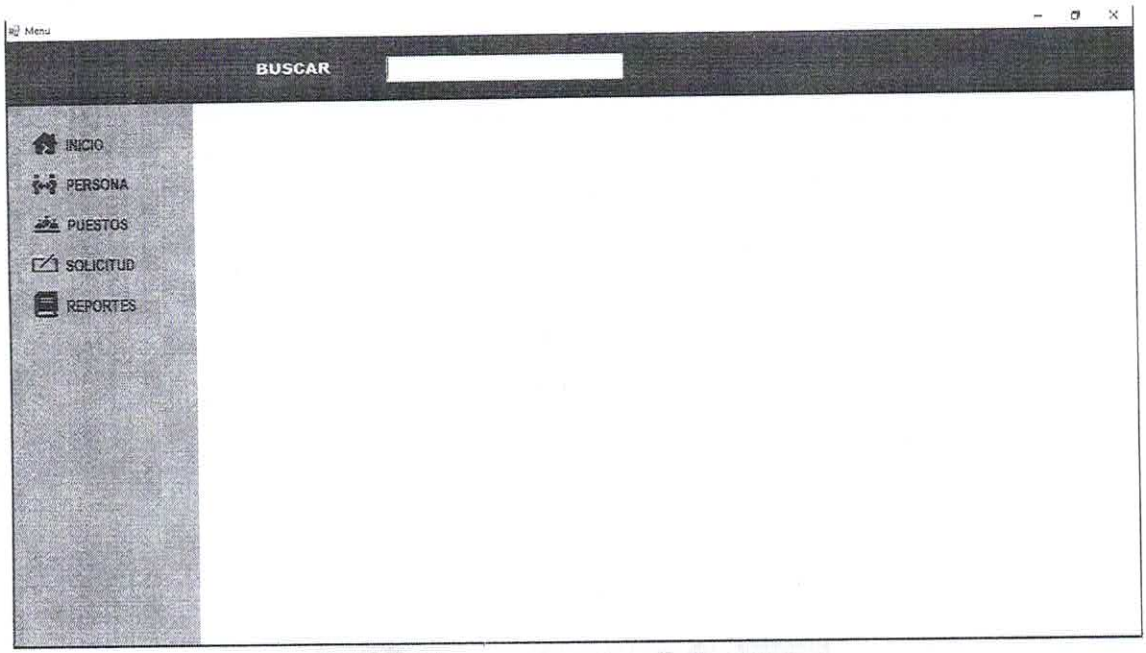
Ejecución de avances del Sistema.

Login. Aquí Configuramos nuestro usuario de pruebas de ejecución.



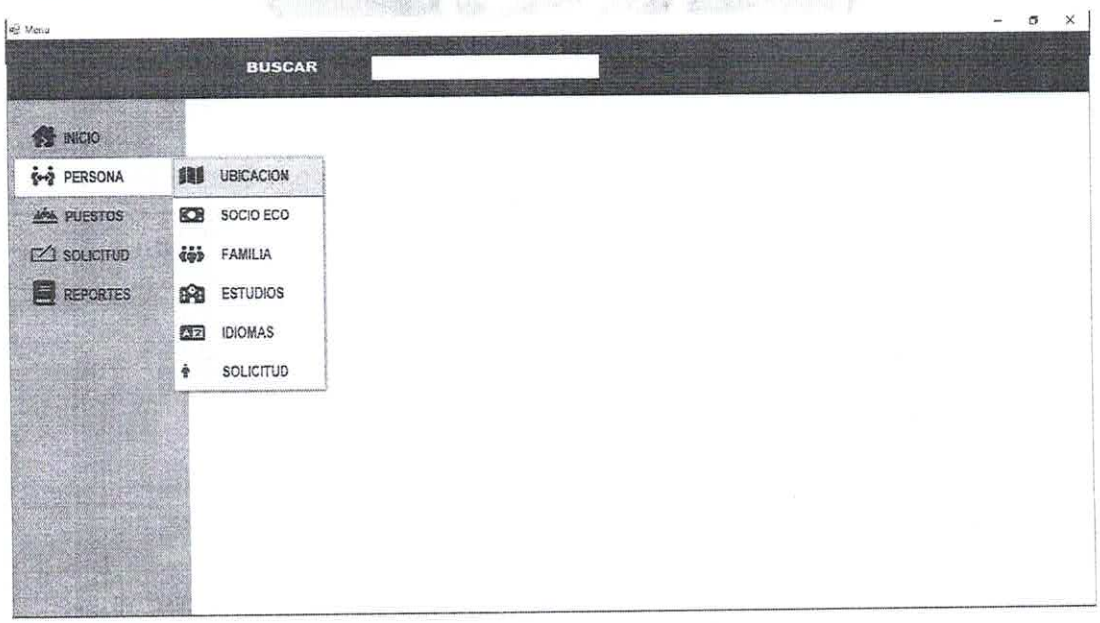
Al tener ingresado nuestras credenciales damos clic en el botón de Acceder.

Al ingresar nos mostrara un menú.

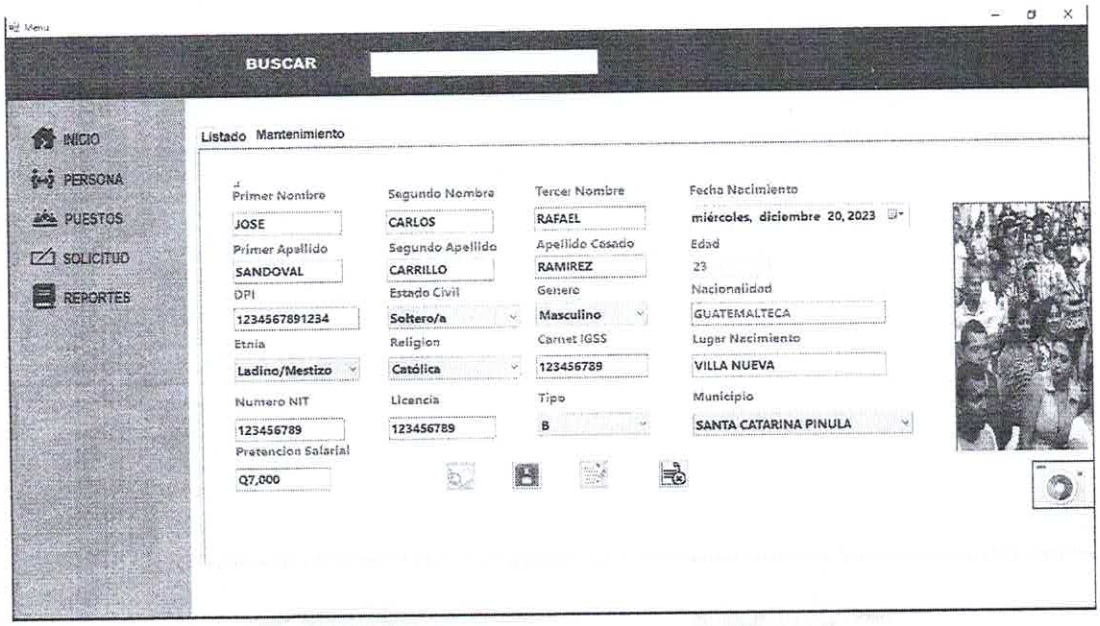


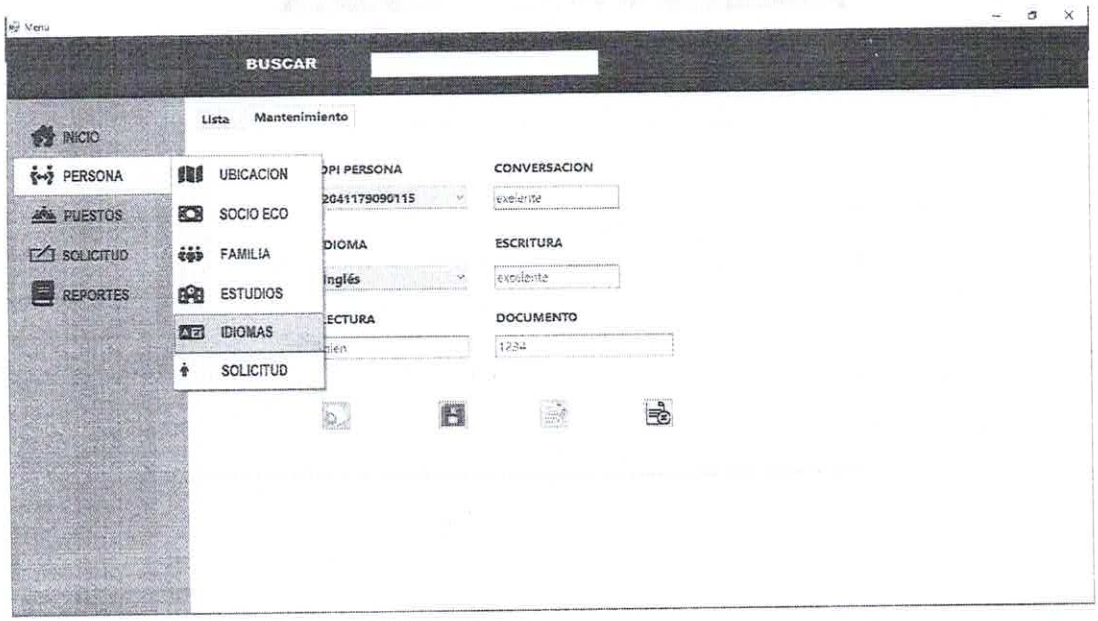
Este menú se compone de las siguientes vistas, una de Inicio aquí se agregarán ciertos datos en el siguiente avance.

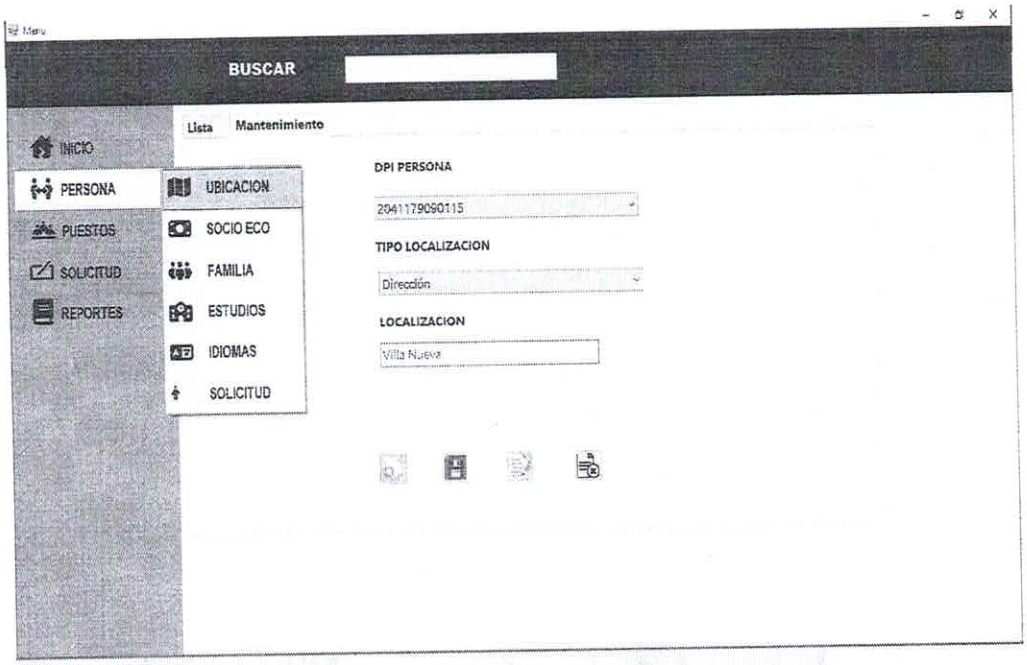
Luego tenemos una opción que dice persona.



Esta tiene un submenú que nuestra la opción que una persona debe de utilizar para poder ingresar sus datos personales y datos generales. Para llenar información de las solicitudes de trabajo.

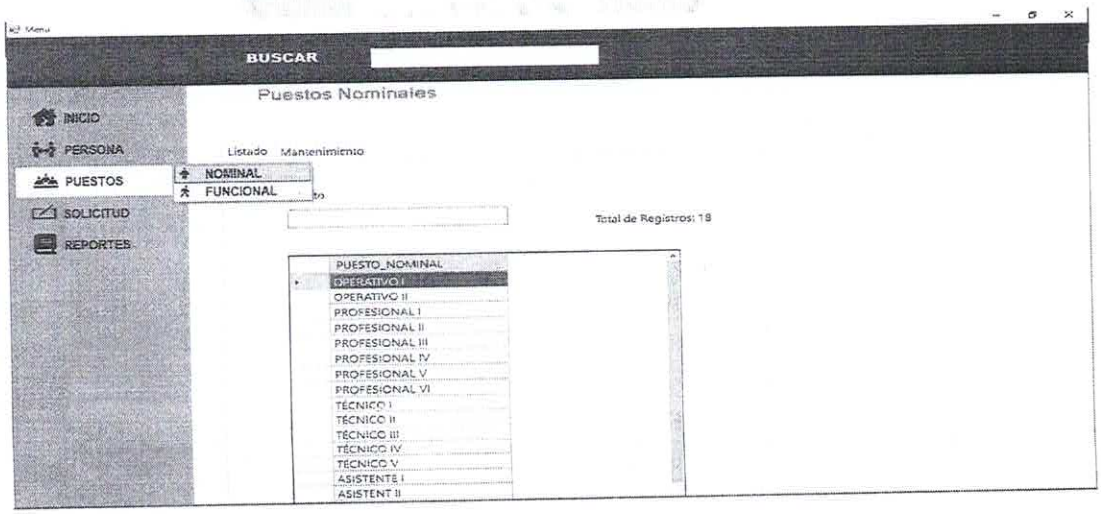






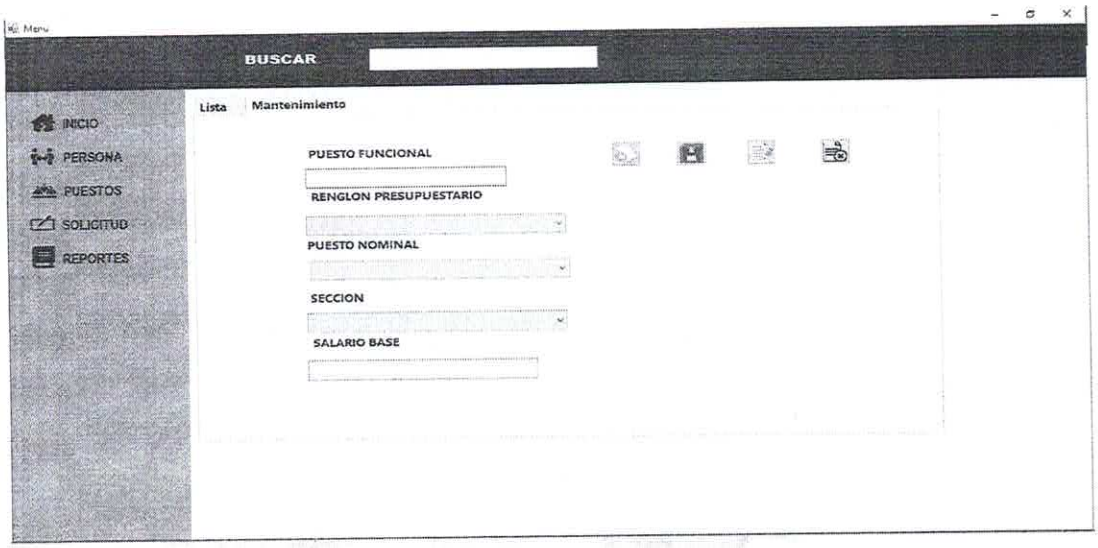
En resumen, la persona se registra con sus datos, luego va ingresar información general como estudio, Idiomas que maneja, Localizaciones, Su Familia, etc esto para poder llenar la Solicitud de trabajo.

Luego en el menú siguen los puestos.



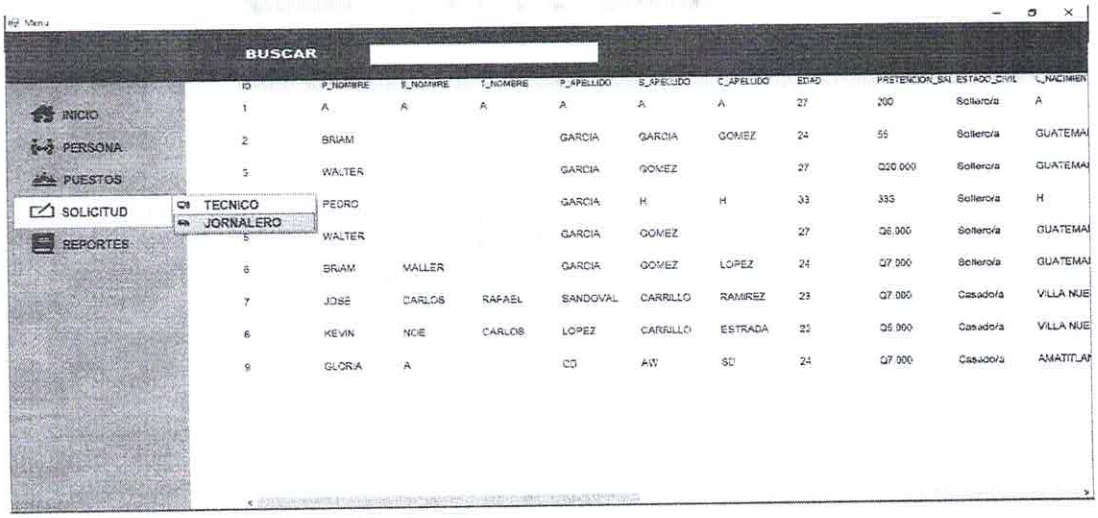
Aquí van estar los tipos de puestos Nominales que se manejan en la ENCA.

Y Los también los puestos Funcionales.



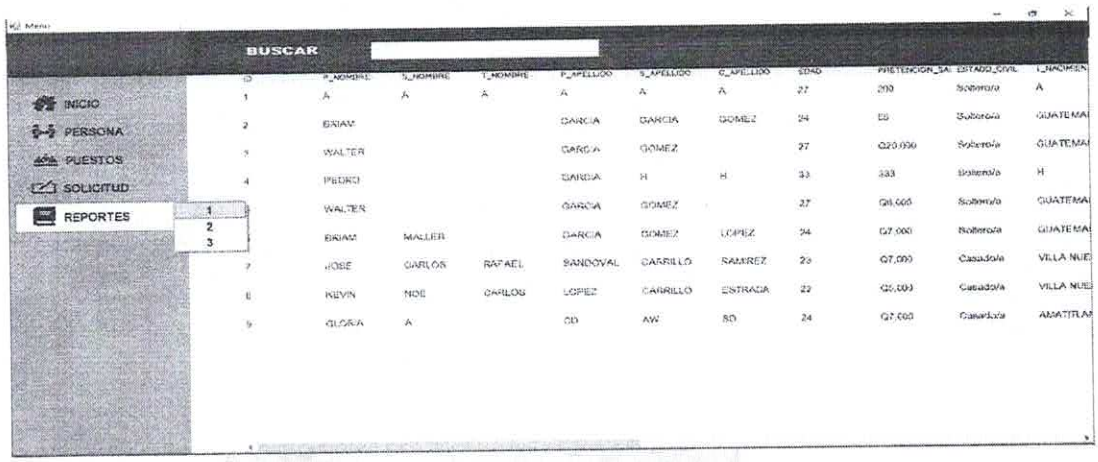
Este va manejar los renglones presupuestarios, los puestos nominales y la sección a la que va ser asignado.

Luego siguen las solicitudes.



En esta parte se harán las solicitudes con todos los datos que se ingresaron de la persona.

Y por último los reportes. Que se configuran en el siguiente avance donde se tenga toda la información y las solicitudes para hacer nuestros reportes.



Conclusión.

En este proyecto, se ha desarrollado una sólida arquitectura de capas en una aplicación de Windows Forms utilizando el lenguaje de programación C# y una base de datos SQL Server. La estructura de capas proporciona una clara separación entre la interfaz de usuario, la lógica de negocios y la capa de acceso a datos, lo que facilita la mantenibilidad, escalabilidad y estabilidad del código.

En la interfaz de usuario, se ha implementado un menú que ofrece funcionalidades relacionadas con datos de personas, solicitudes y reportes. Este enfoque modular permite una fácil navegación entre las diferentes secciones de la aplicación, proporcionando una experiencia de usuario intuitiva y eficiente.

En el siguiente avance del proyecto, se planea incorporar la gestión de solicitudes de empleo y mejorar el diseño general de la aplicación. Este desarrollo futuro sugiere una expansión en la funcionalidad de la aplicación, lo que podría incluir la creación, visualización y gestión de solicitudes de empleo. Además, se explorarán mejoras en el diseño de la interfaz de usuario para proporcionar una experiencia más atractiva y fácil de usar.

En cuanto a la base de datos, se ha utilizado SQL Server para almacenar y gestionar los datos de la aplicación. En futuros avances, se considerará la optimización de consultas, la normalización de la base de datos y la implementación de medidas de seguridad adicionales para garantizar la integridad y confidencialidad de los datos.

En resumen, la aplicación actual representa un sólido punto de partida con una arquitectura bien definida y funcionalidades básicas implementadas. Con los avances

planeados, se espera mejorar la aplicación en términos de funcionalidad, diseño y rendimiento, asegurando así una experiencia de usuario más completa y satisfactoria.

Referencias.

https://www.videojuegosydesarrollo.com/wp-content/uploads/2019/09/modelo.fw_.png

https://www.studocu.com/pe/document/universidad-peruana-de-ciencias-aplicadas/programacion-web/programacion-por-capas-en-visual-studio-c/32470206

https://iconos8.es/icons/set/windows-forms

https://learn.microsoft.com/es-es/sql/relational-databases/stored-procedures/create-a-stored-procedure?view=sql-server-ver16