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: Marzo-2024

f. _____

EPSista. Briam Garcia Gómez.

f. _____

Ing. Gloria Ercira Colindres Solórzano.

Encargada de la unidad de Informática.

f. _____

Ing. Jorge Escobar

Sub Director ENCA.

Marzo-2024

Informe de Avances Sistema Recursos Humanos.

EPSista. Briam Garcia Gómez.

Índice.

Introducción.

El diseño de pantallas en Windows Form en C# es crucial para proporcionar una experiencia de usuario fluida y atractiva. Al crear un módulo de usuario con diferentes roles (administrador, usuario operativo del sistema), es esencial garantizar la accesibilidad y la eficiencia en la navegación y la interacción.

Para lograr un diseño efectivo, se deben tener en cuenta los principios de usabilidad, como la organización intuitiva de la información, la claridad en la presentación de datos y la consistencia en la interfaz. Esto puede lograrse mediante la disposición lógica de los elementos, el uso adecuado de colores y tipografías, y la incorporación de controles visuales que guíen al usuario a través del sistema de manera coherente.

Además del diseño visual, es importante desarrollar funcionalidades que permitan la gestión eficiente de los datos. En el caso de la creación de reportes del sistema, se deben implementar procesos que recopilen y procesen la información ingresada por los usuarios, generando informes relevantes y personalizables según las necesidades del usuario.

Para el módulo de vacaciones, se deben proporcionar opciones para agregar periodos vacacionales, insertar solicitudes de vacaciones dentro de esos periodos y cancelar solicitudes existentes. Esto implica la creación de formularios de entrada de datos y la implementación de lógica de negocio para gestionar estas operaciones de manera segura y eficiente.

Diseño de pantallas y menú.

Para el diseño de la interfaz de usuario en el desarrollo de Windows form y programación en C# se deben de tener en cuenta las siguientes consideraciones.

Diseño visual atractivo: Aunque Windows Forms no ofrece tantas opciones de diseño visual como otras tecnologías más modernas, aún puedes crear interfaces atractivas utilizando colores, fuentes y disposiciones de manera efectiva. Mantén un diseño limpio y coherente en todas las pantallas para una experiencia de usuario uniforme.

Organización lógica de controles: Agrupa los controles relacionados de manera lógica y coherente en tus formularios. Por ejemplo, coloca los campos de entrada relacionados cerca uno del otro y utiliza etiquetas descriptivas para indicar qué información se espera en cada campo.

Validación de entrada de datos: Implementa validación de entrada de datos para garantizar que los usuarios ingresen información válida en los campos correspondientes. Puedes utilizar eventos como KeyPress o Validating para realizar validaciones en tiempo real y proporcionar retroalimentación instantánea al usuario sobre cualquier error.

Manejo de errores: Anticipa posibles errores de entrada de datos o problemas de ejecución y proporciona mensajes de error claros y descriptivos cuando sea necesario. Esto ayuda a los usuarios a comprender qué salió mal y cómo pueden corregir el problema.

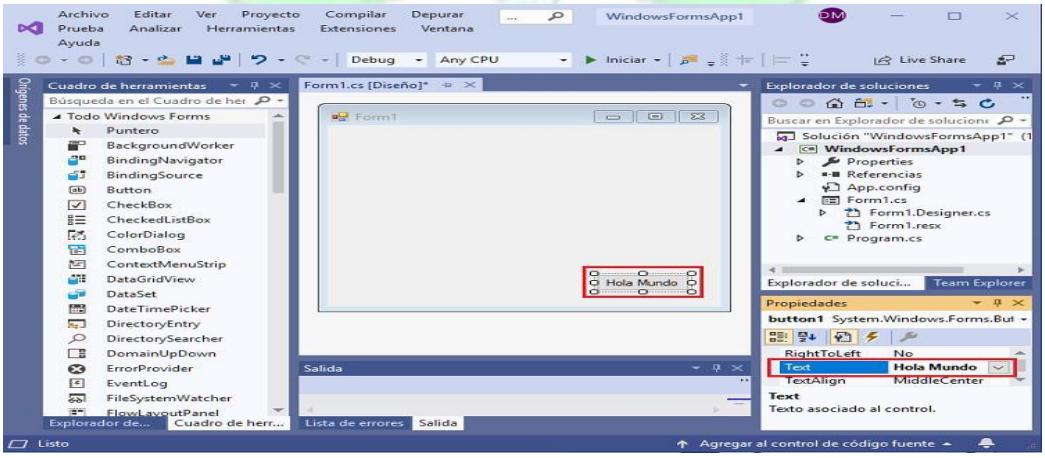
Interactividad: Aprovecha al máximo la interactividad que ofrece Windows Forms mediante el uso de controles como botones, casillas de verificación, listas desplegables

y cuadros de diálogo para mejorar la experiencia del usuario y facilitar la navegación por tu aplicación.

Usabilidad: Prioriza la usabilidad al diseñar tus interfaces de usuario. Haz que las acciones comunes sean fáciles de realizar y minimiza la cantidad de clics o pasos necesarios para completar tareas importantes.

Pruebas de usuario: Al igual que en cualquier proyecto de desarrollo de software, realiza pruebas exhaustivas de tus interfaces de usuario con usuarios reales para identificar posibles problemas de usabilidad y realizar ajustes según sea necesario.

En resumen, aunque Windows Forms puede ser menos flexible en términos de diseño visual en comparación con otras tecnologías más modernas, aún puedes crear interfaces de usuario efectivas y atractivas siguiendo buenas prácticas de diseño y usabilidad en base a esto se diseñó las siguientes interfaces para el funcionamiento del sistema.

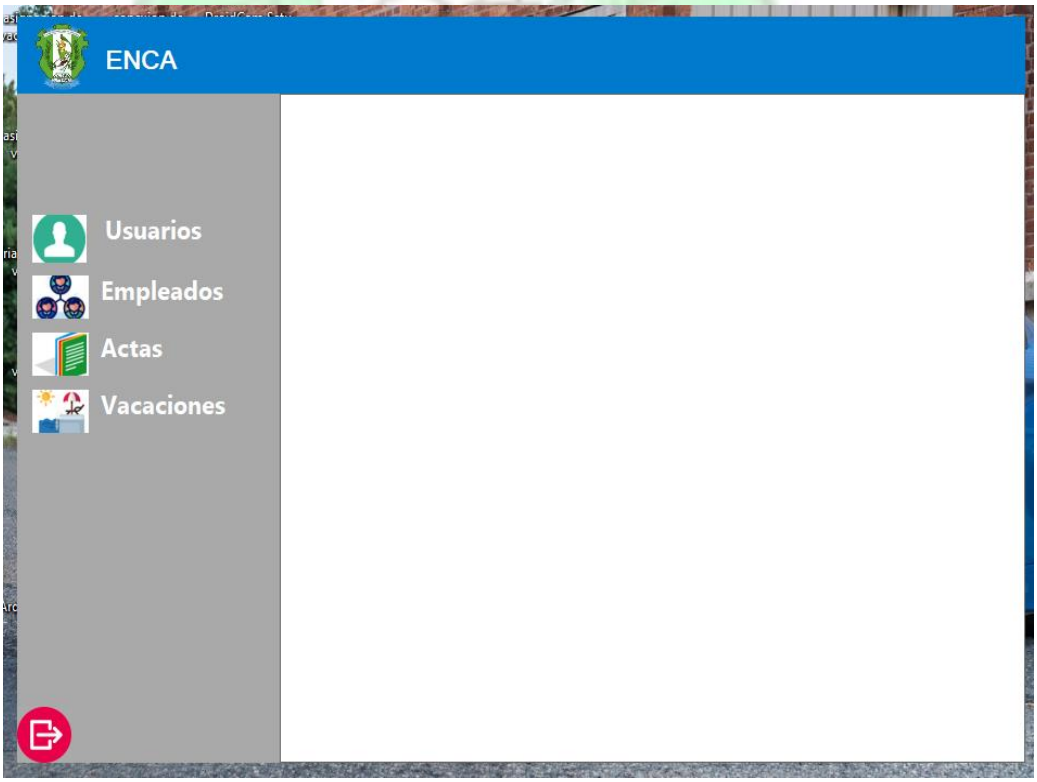


Interfaz de Login:

La Interfaz de Login del Sistema para validación de Usuario.

Interfaz de Menú.

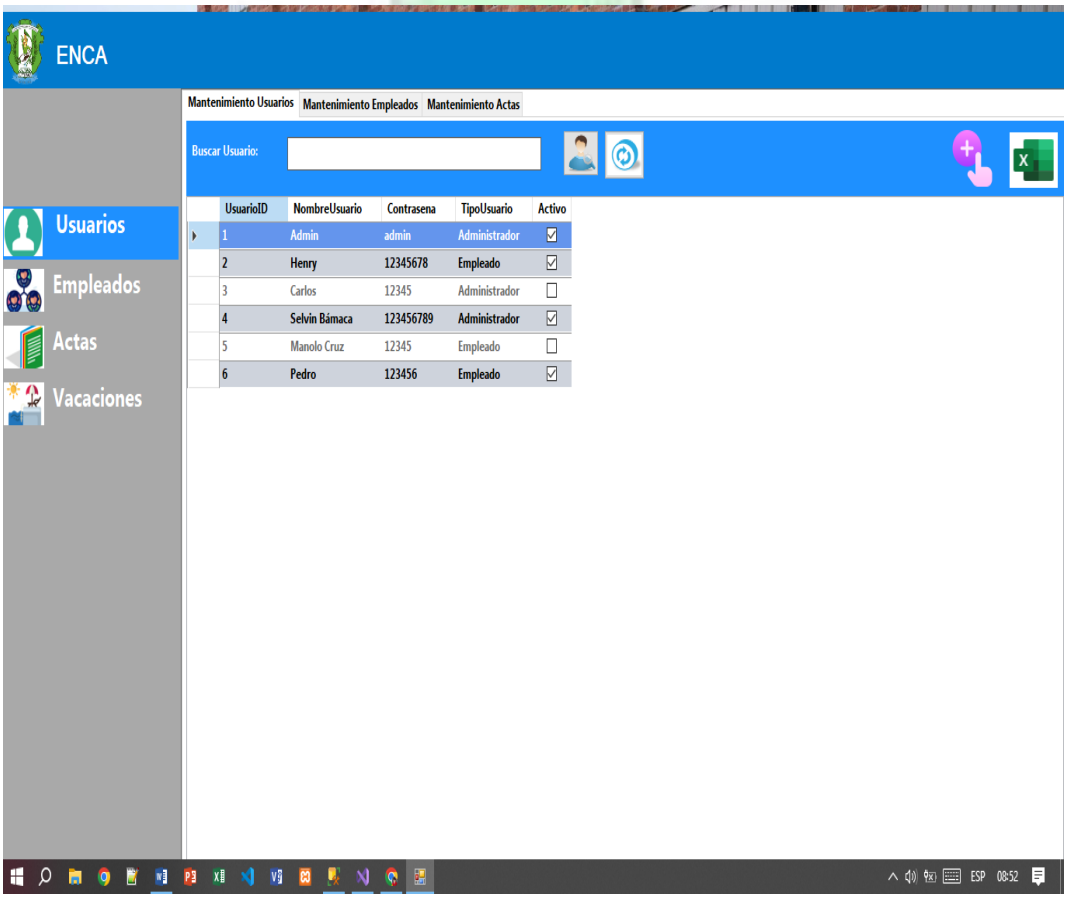
Diseño de menú interfaz maestra. Que controla el inicio del programa y conlleva a todas las funcionalidades del sistema.



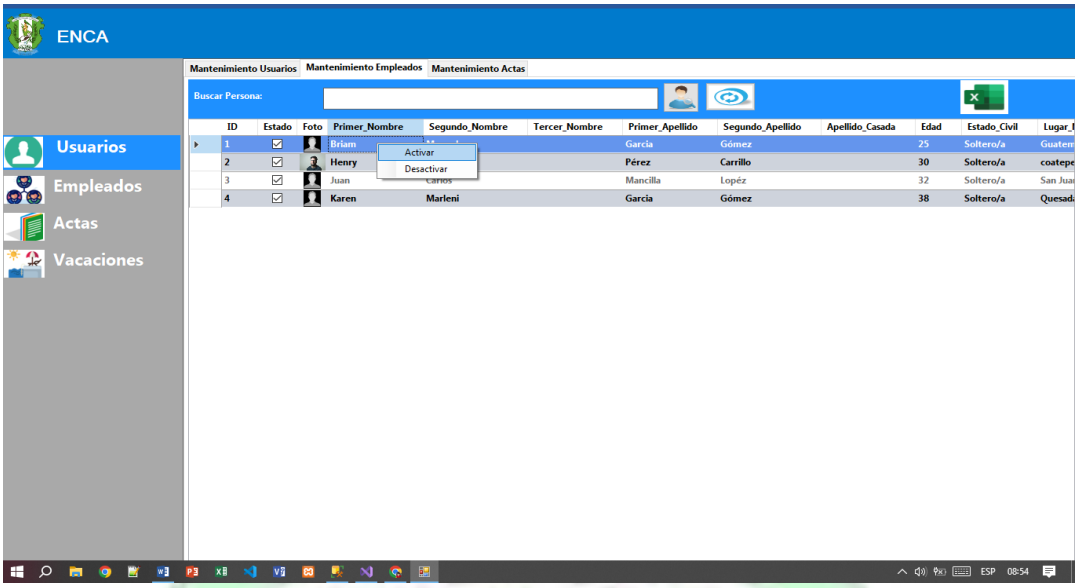
En esta interfaz de menú se tiene una barra superior con el logo de la ENCA, y otra barra a todo el largo de la ventana en el lago izquierdo de la pantalla y una barra que ocupa el resto de la pantalla. El cual va direccionar las funcionalidades del sistema como vemos hay un menú el cual contiene las funcionalidades del sistema.

En base este menú se verá cada opción del menú.

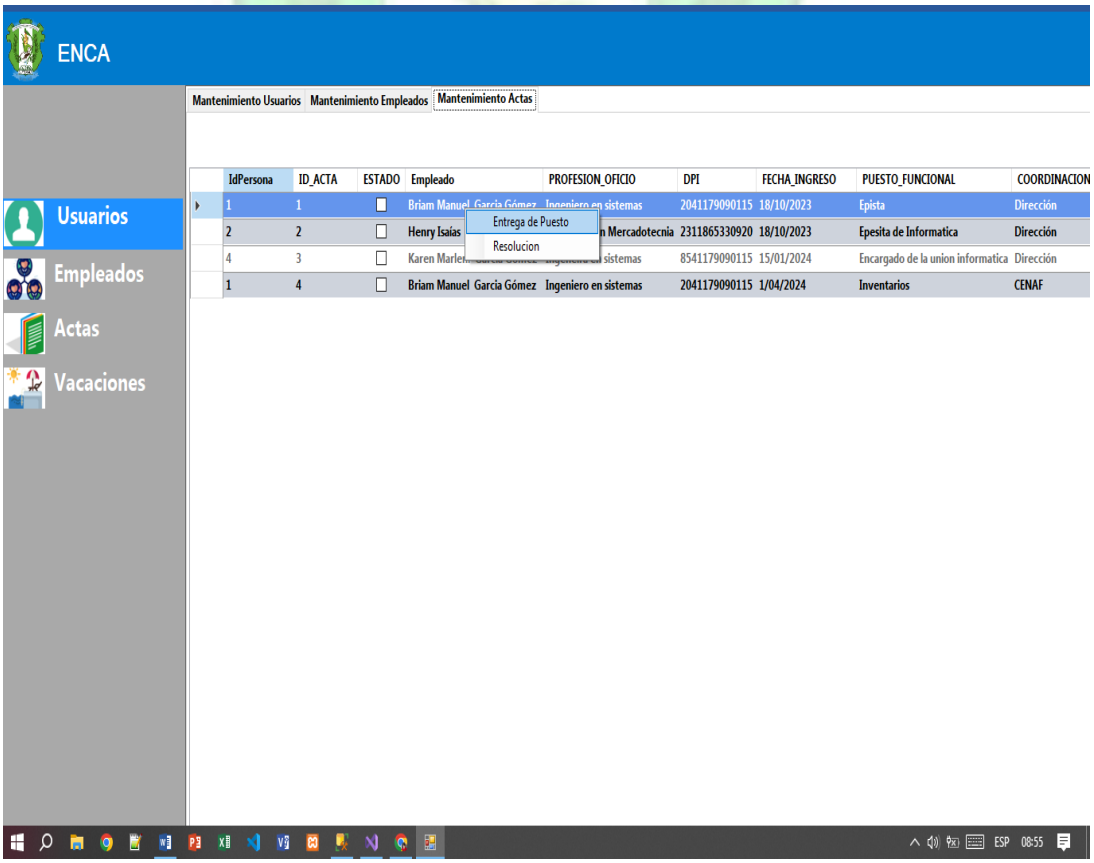
Menú de Usuarios.

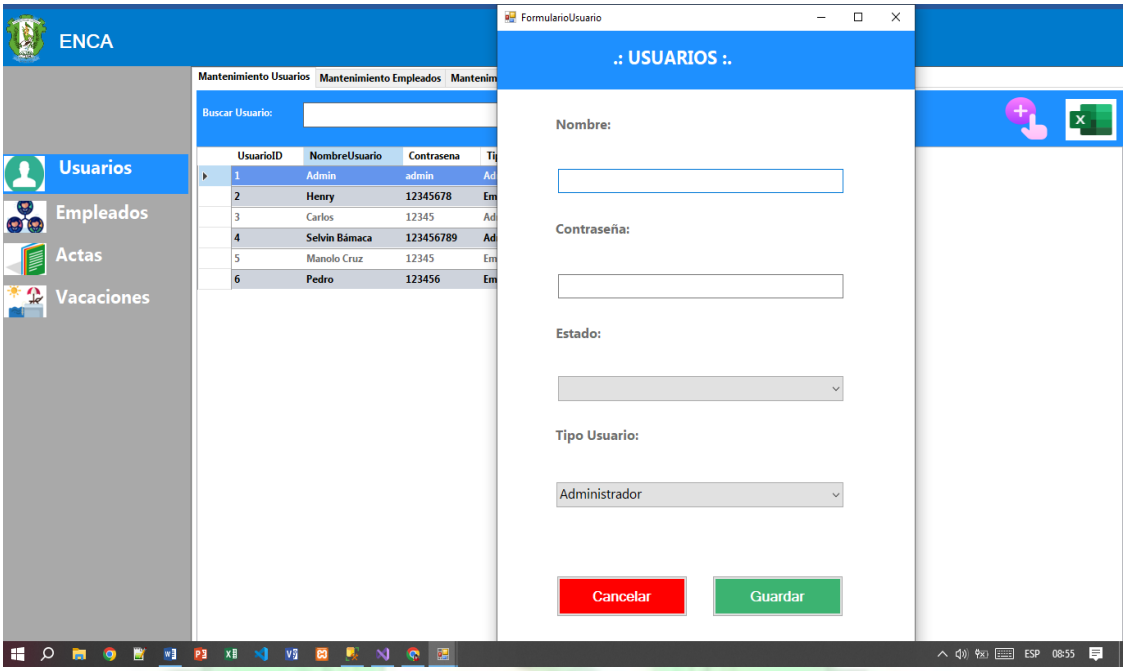


Aquí sé que los usuarios que se han creado y también se tiene diferentes funcionalidades para administrar a los empleados y las actas de entregas de puestos.



Como activar o desactivar un empleado, búsquedas y hacer reportes y en Exceles con los datos de los empleados y usuarios del sistema.

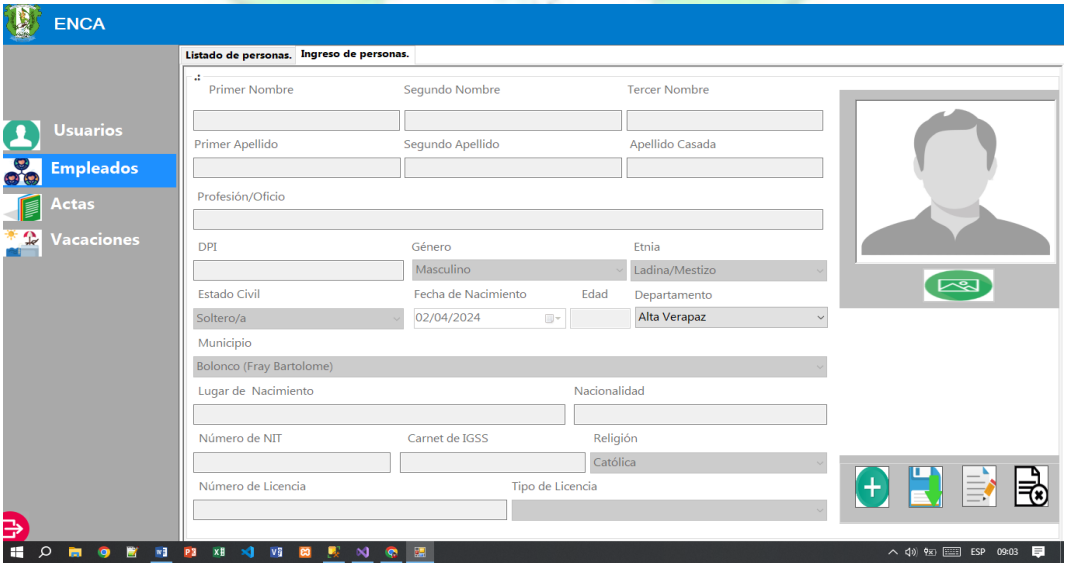




Y también crear nuevo usuario.

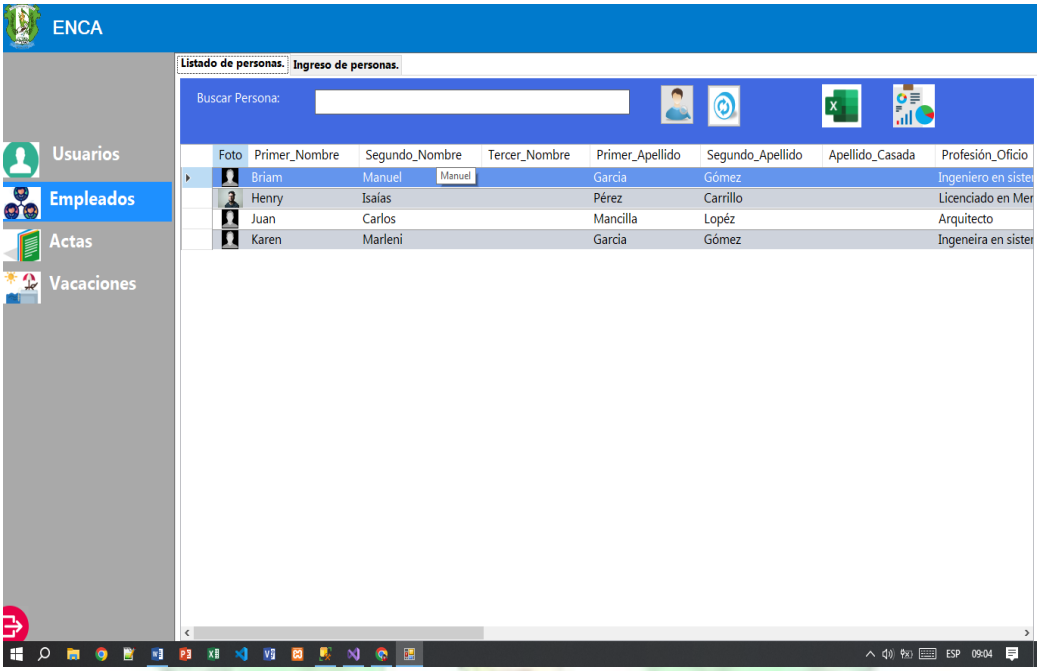
Menú de Empleados.

En el menú de empleados se puede ingresar un empleado con todos sus datos.

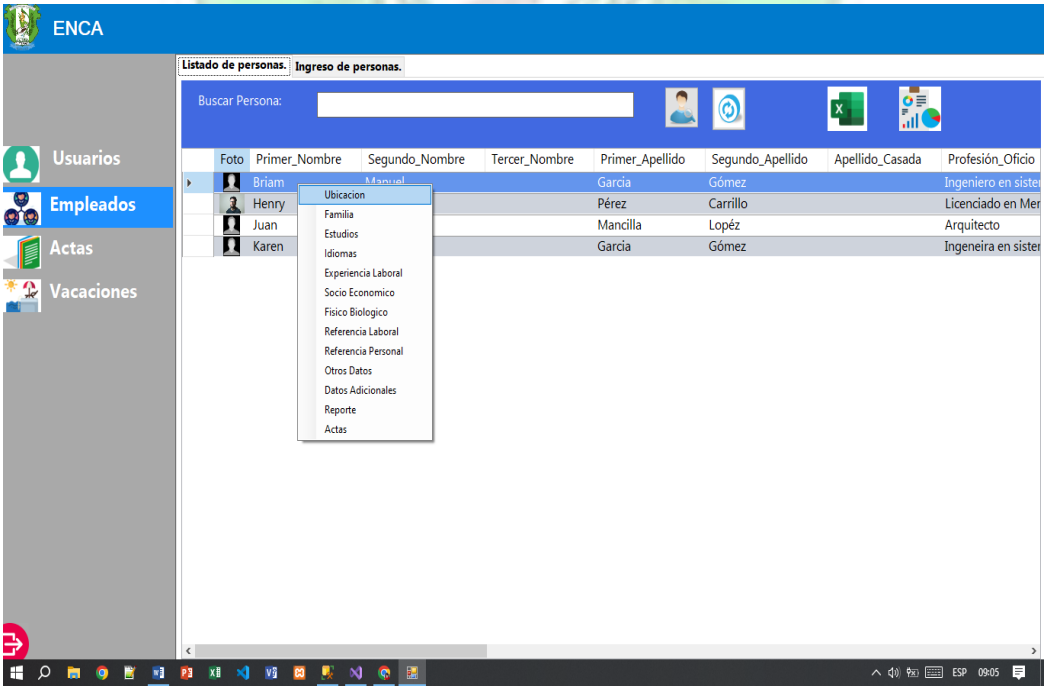


Este es el formulario de ingreso de los datos de un empleado.

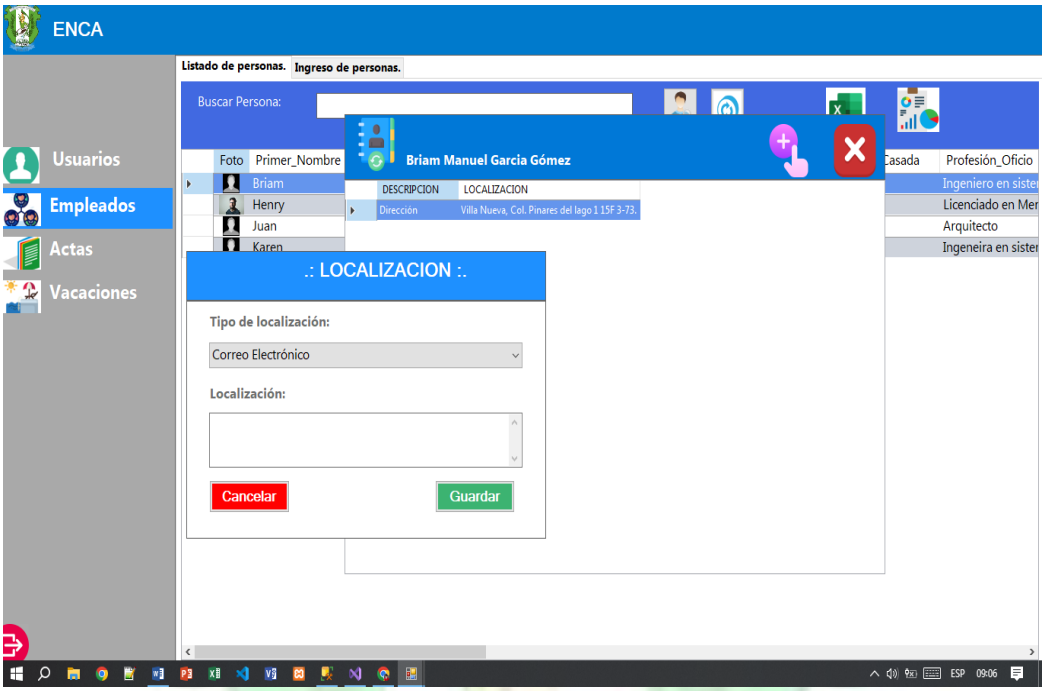
Luego al ingresar el empleado nuestro una lista de empleados.



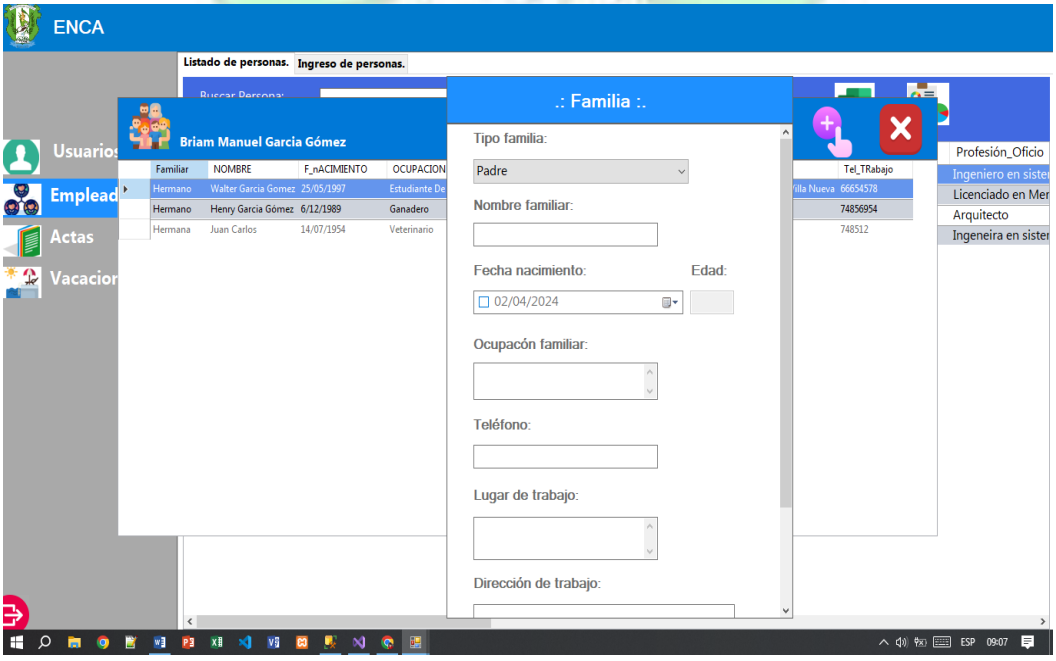
Con toda su información y al seleccionar un empleado en específico se puede ingresar toda su información.



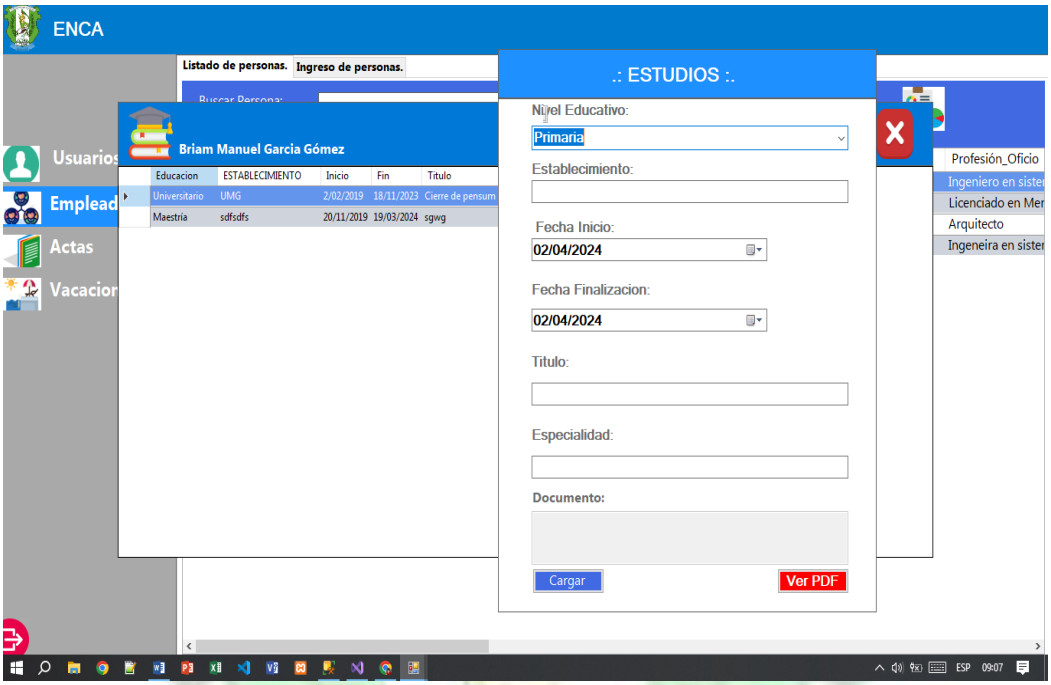
Como ubicaciones.



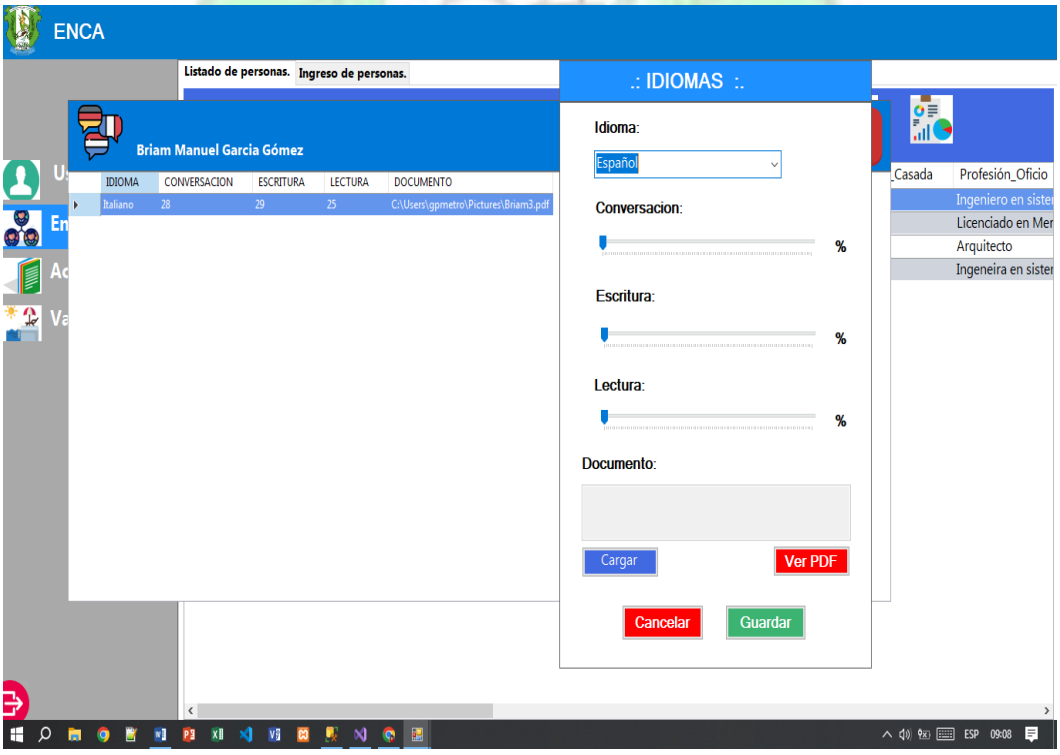
Familiares.



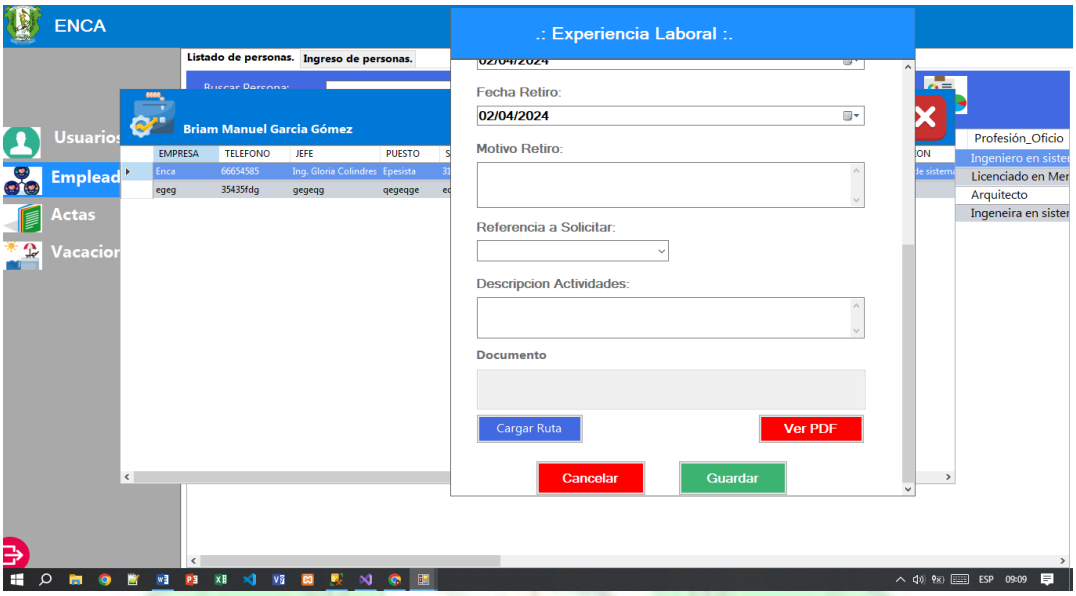
Estudios.



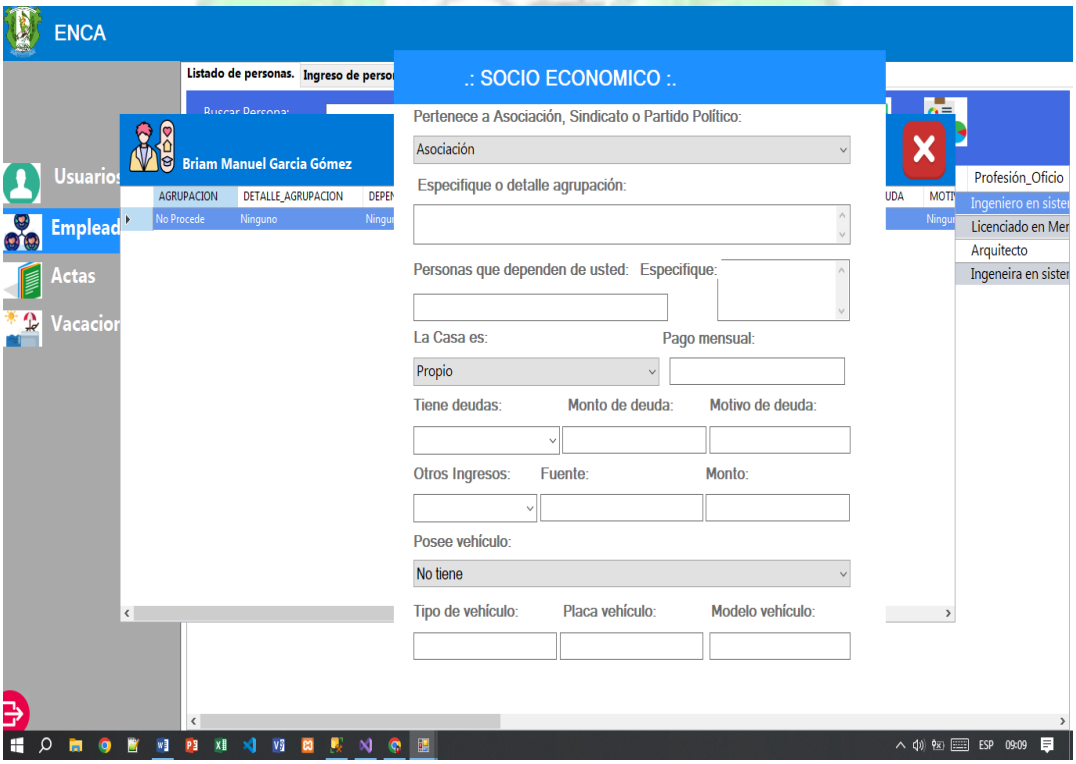
Idiomas.



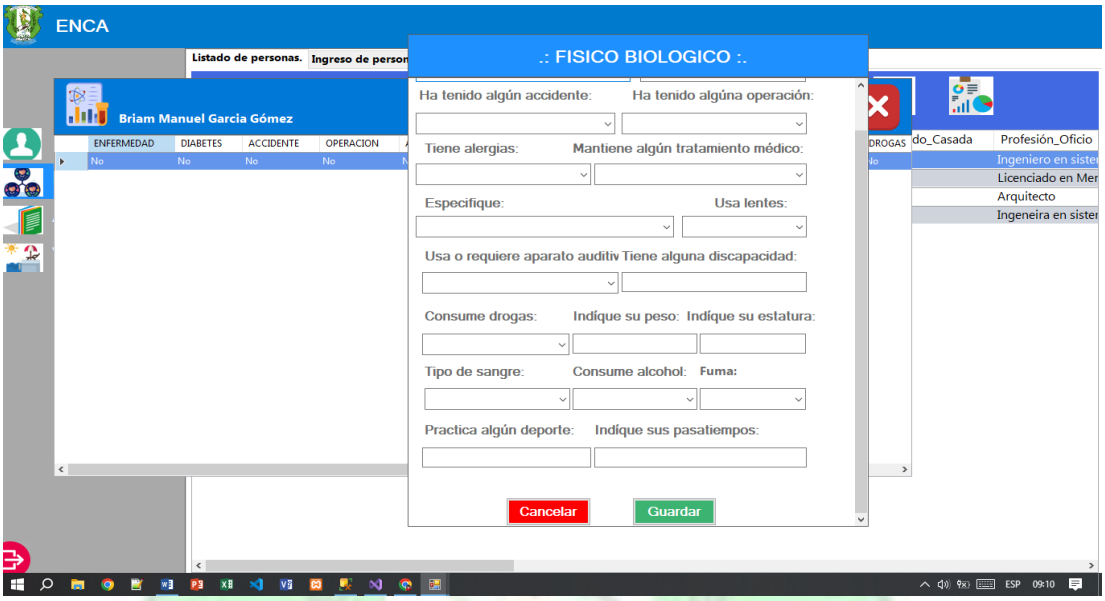
Experiencias laborales.



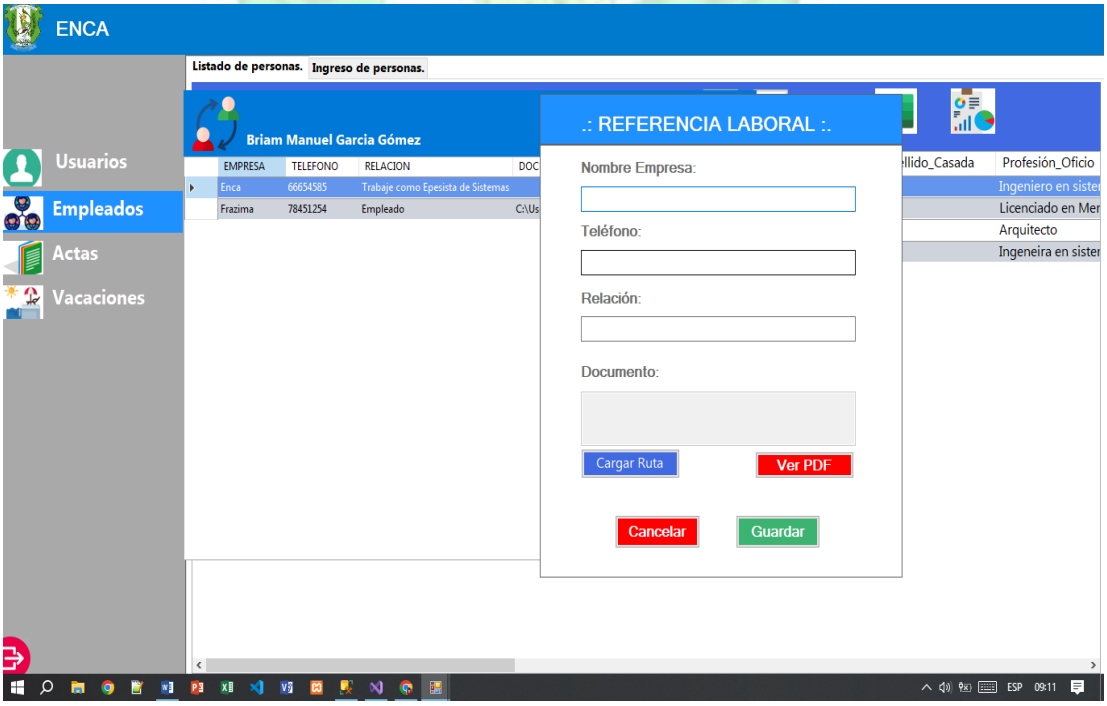
Información socioeconómica.



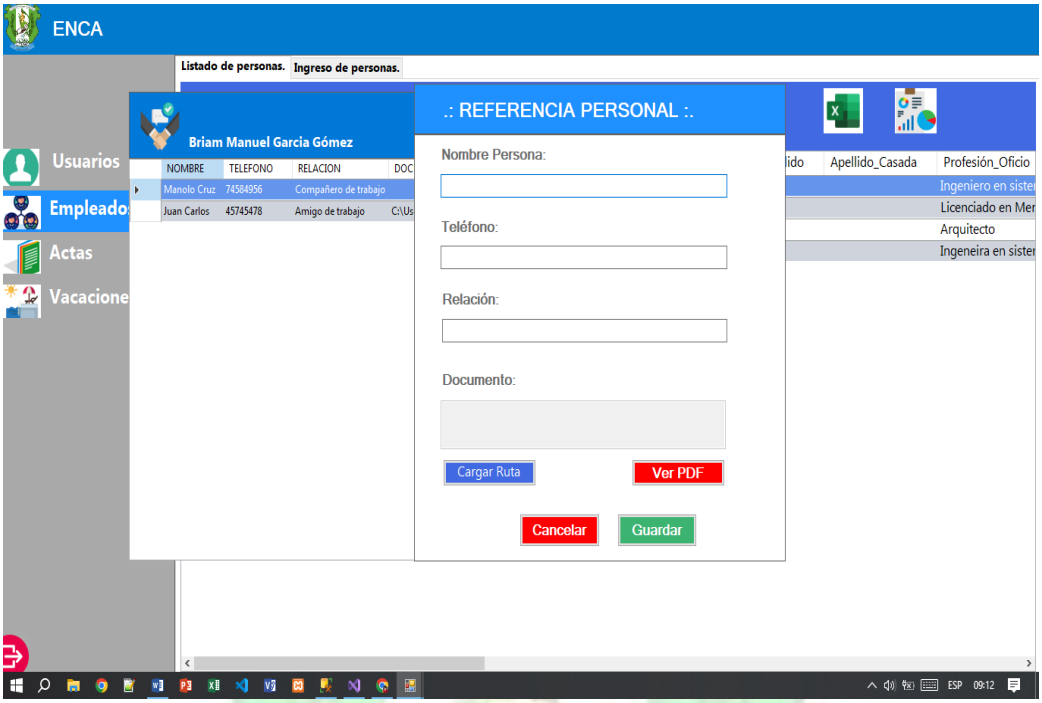
Información física Biológica.



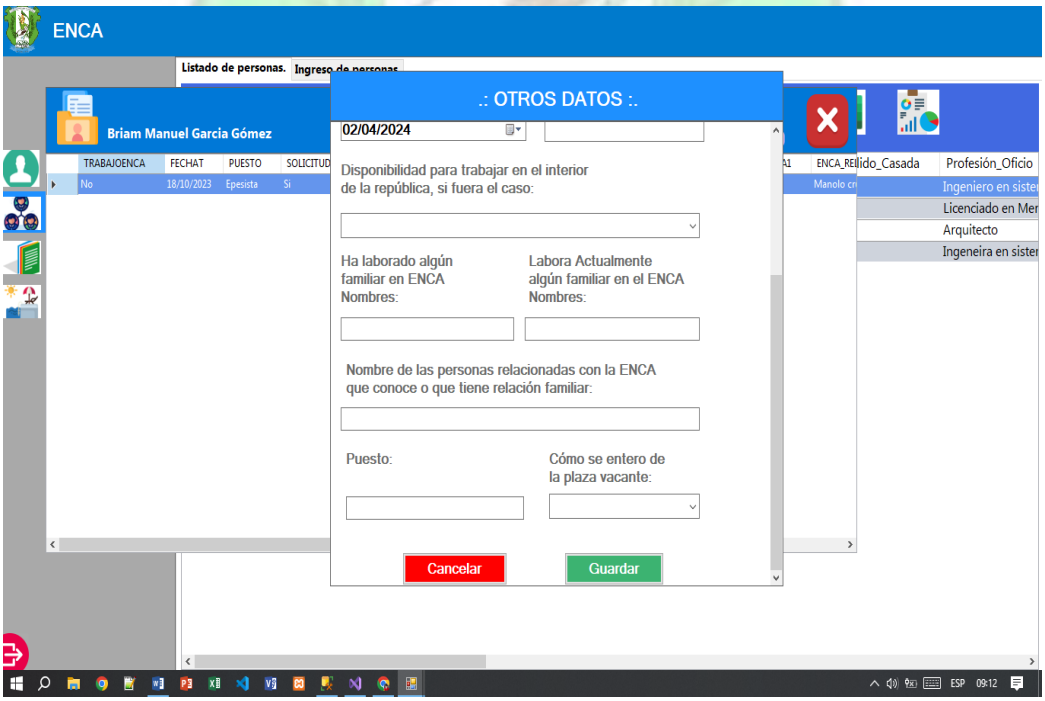
Referencias Laborales.



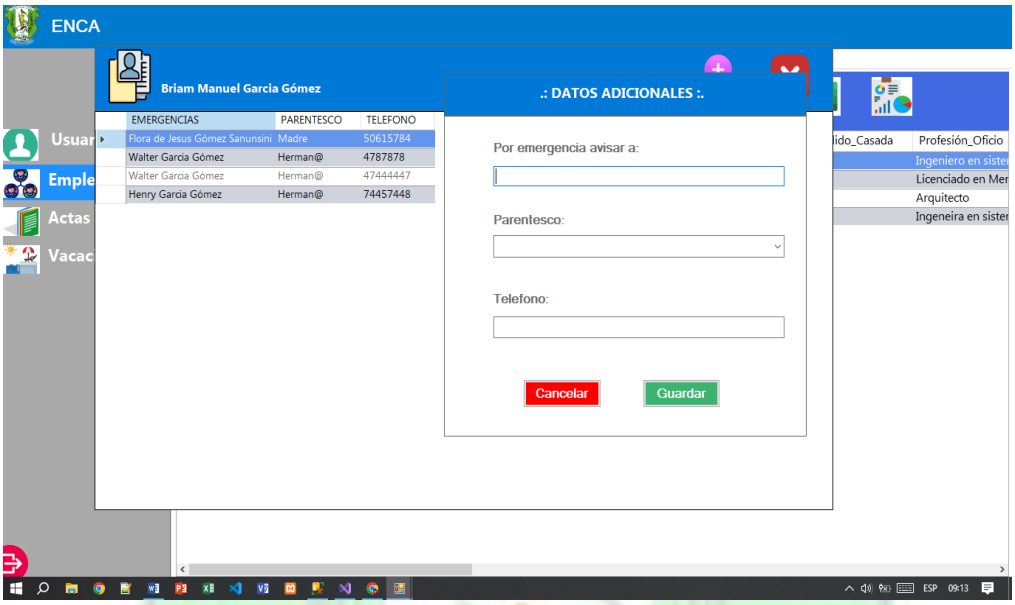
Referencias Personales.



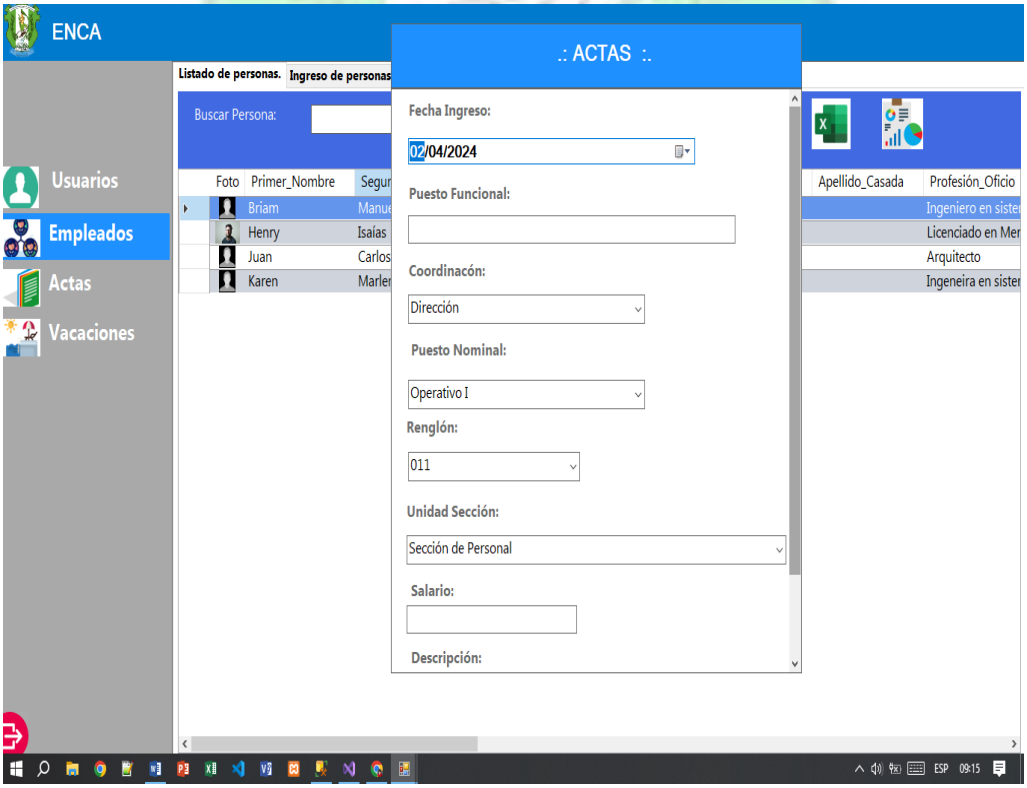
Otros datos del empleado relevantes.



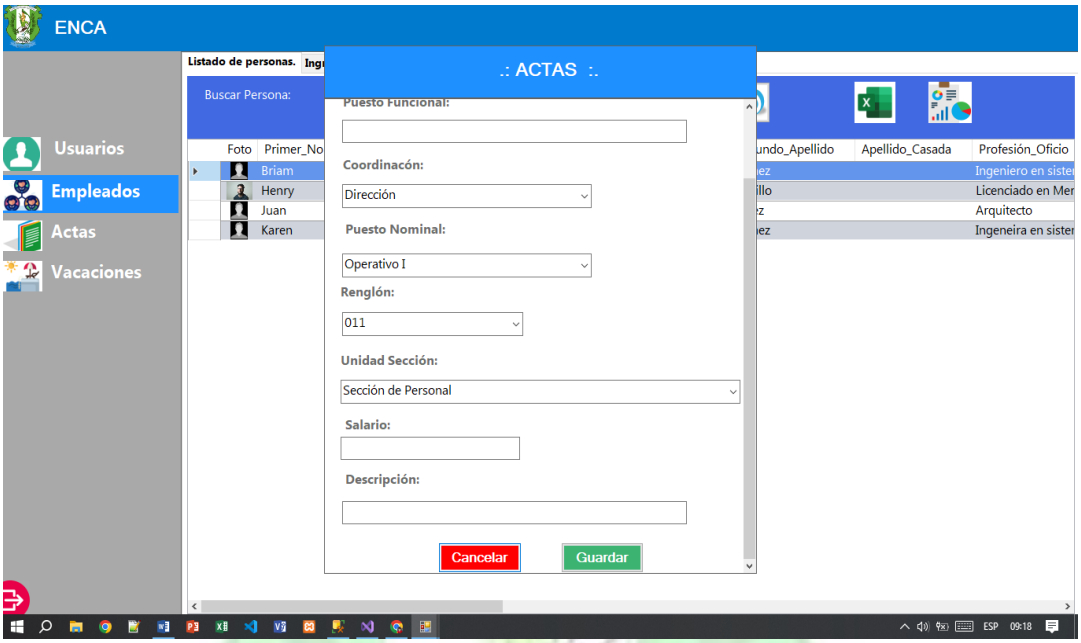
Datos Adicionales del empleado.



Y algo muy importante se hace las actas del empleado. Esto en base a ciertos reglamentos de la Institución ENCA.

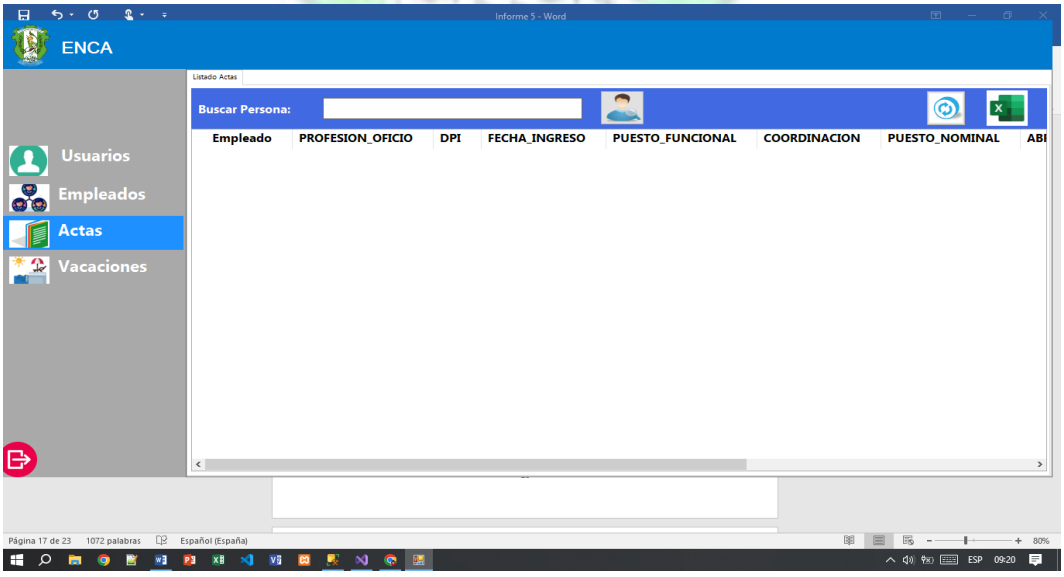


Las cuales son fecha de ingreso, puesto funcional, puesto nominal, coordinación, unidad/sección, renglón, salario base y una descripción breve en base al puesto funcional. Esto para poder hacer el acta de entrega de puesto.

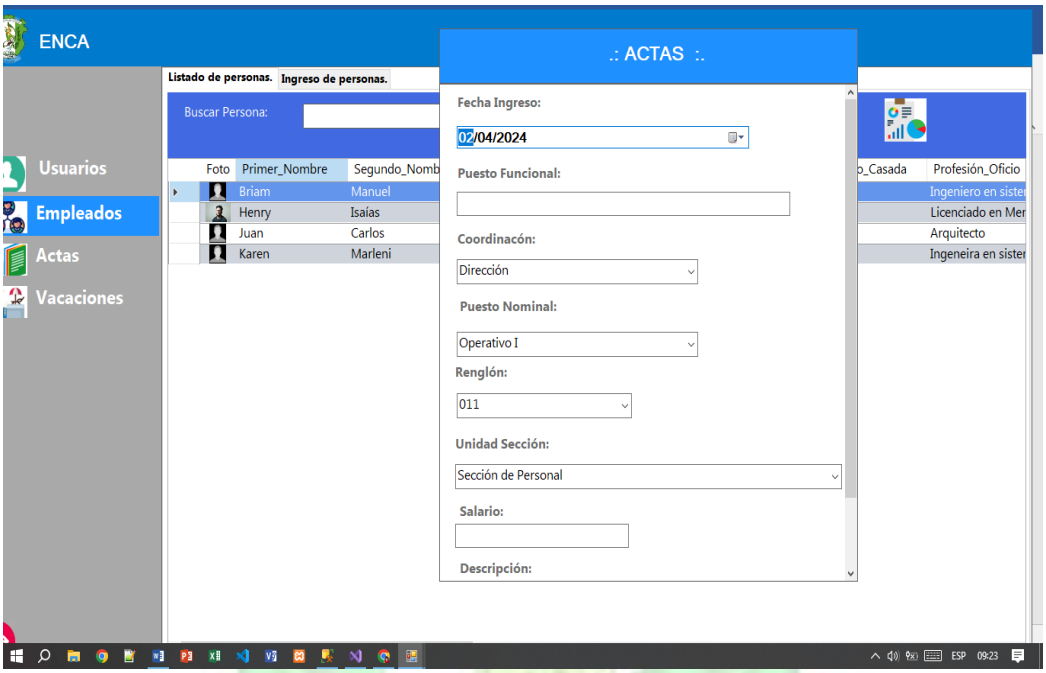


Y esto es en términos generales lo que hace el menú de empleados.

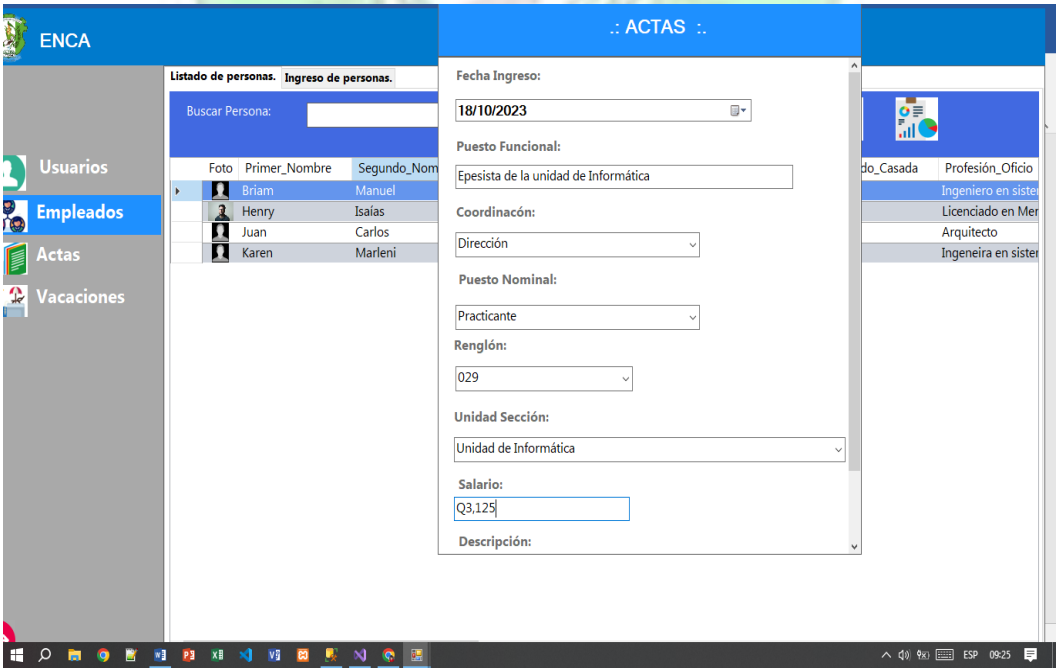
Menú de Actas.

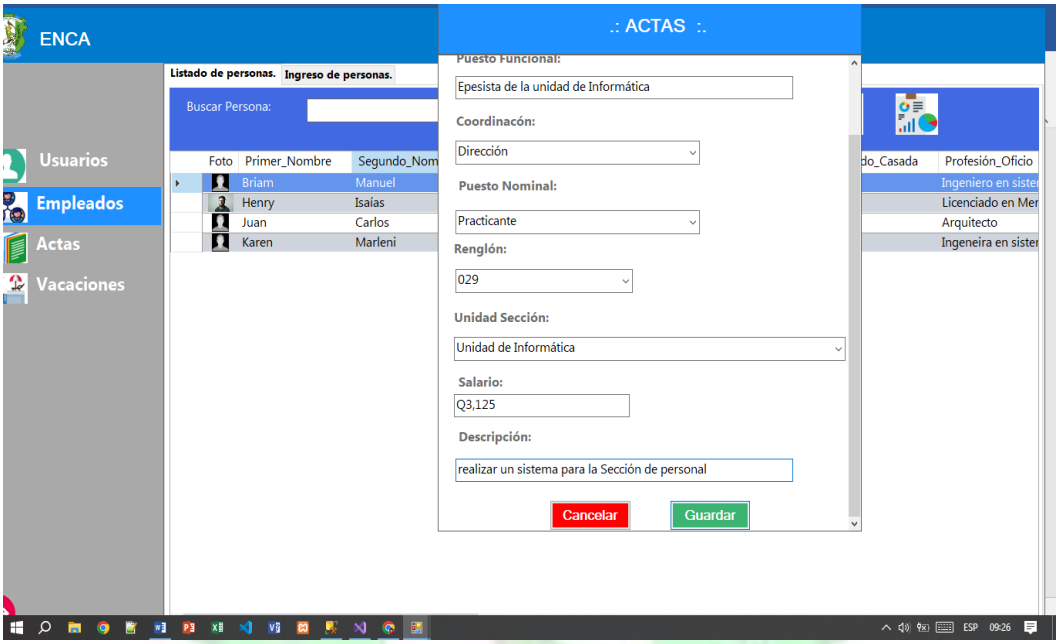


Para lo que son las actas se tiene siguiente lógica desde el menú de empleados se elige a una persona a la cual se le va realizar su acta.

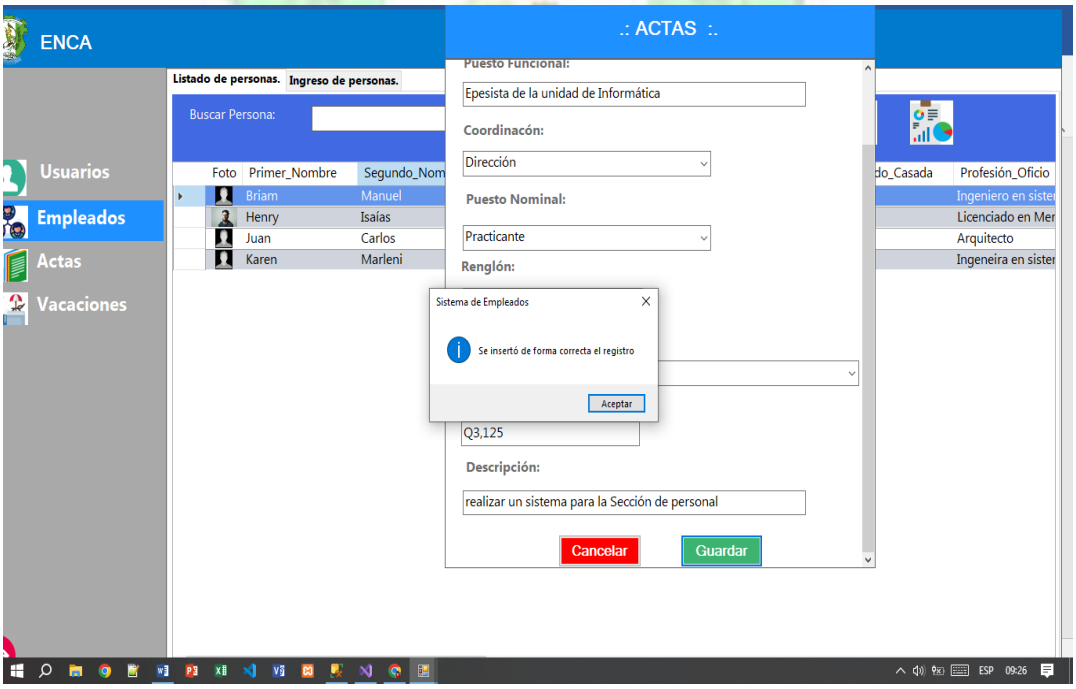


En este caso a Briam, se ingresan los datos del acta.



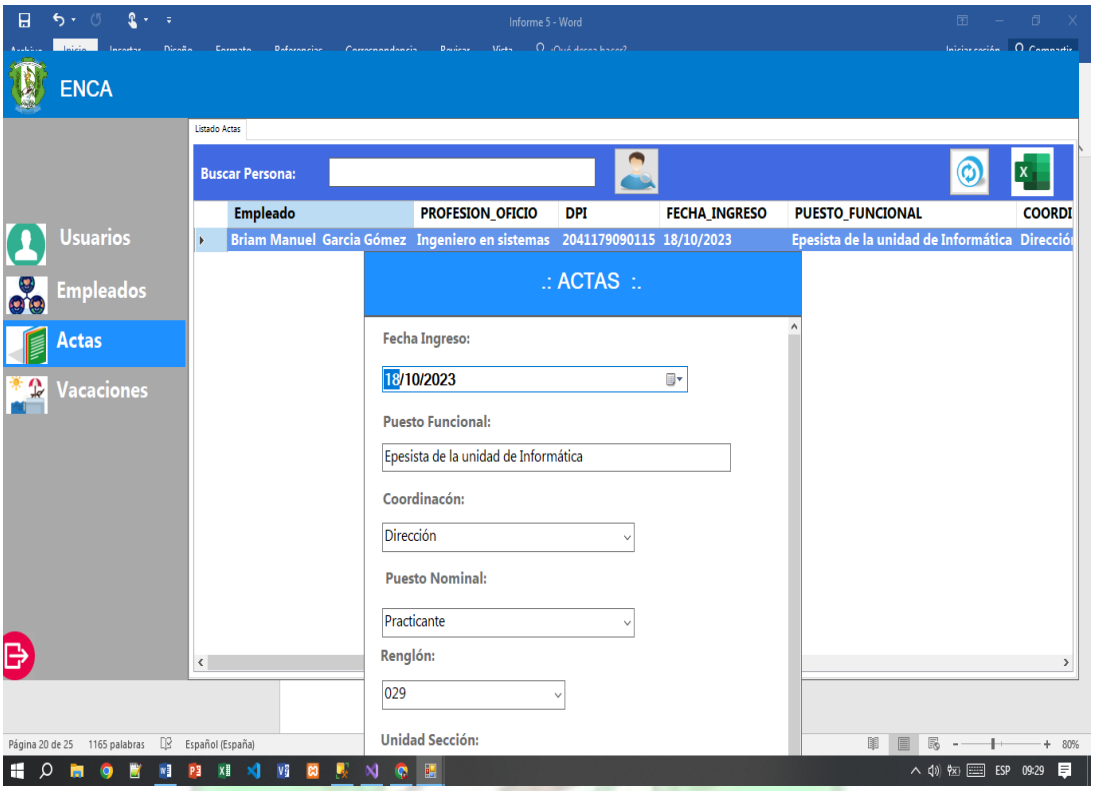


Asignamos los datos y guardamos.



Nos confirma el ingreso del acta en base al empleado y su información correspondiente.

Ya con este ingreso del acta nos dirigimos al menú de actas el cual tendrá el registro del empleado con su información.

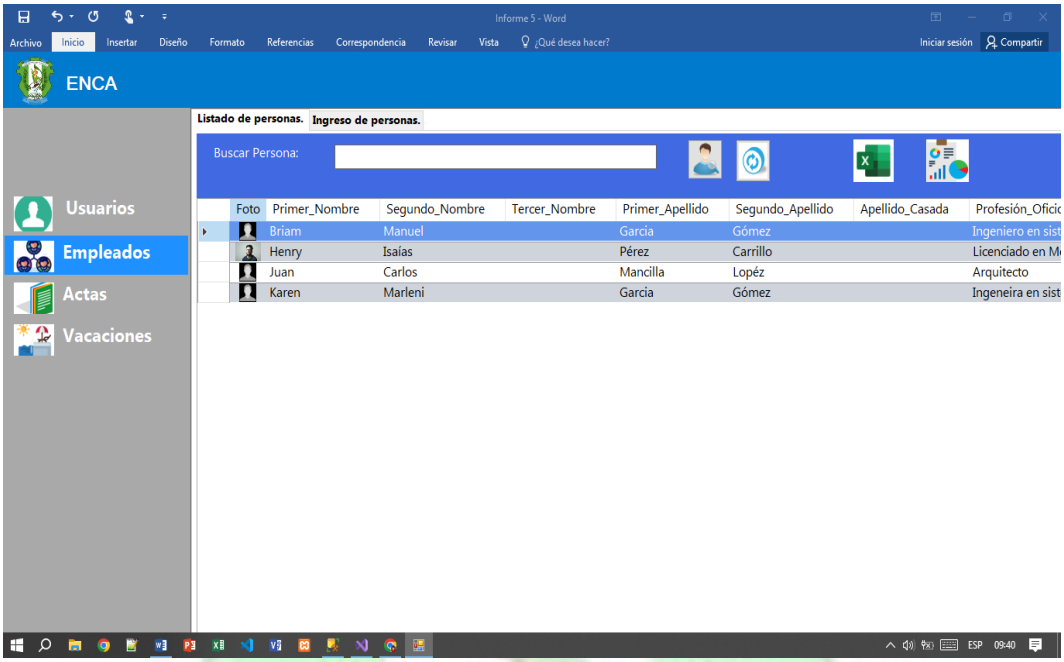


Por si se hace algún cambio.

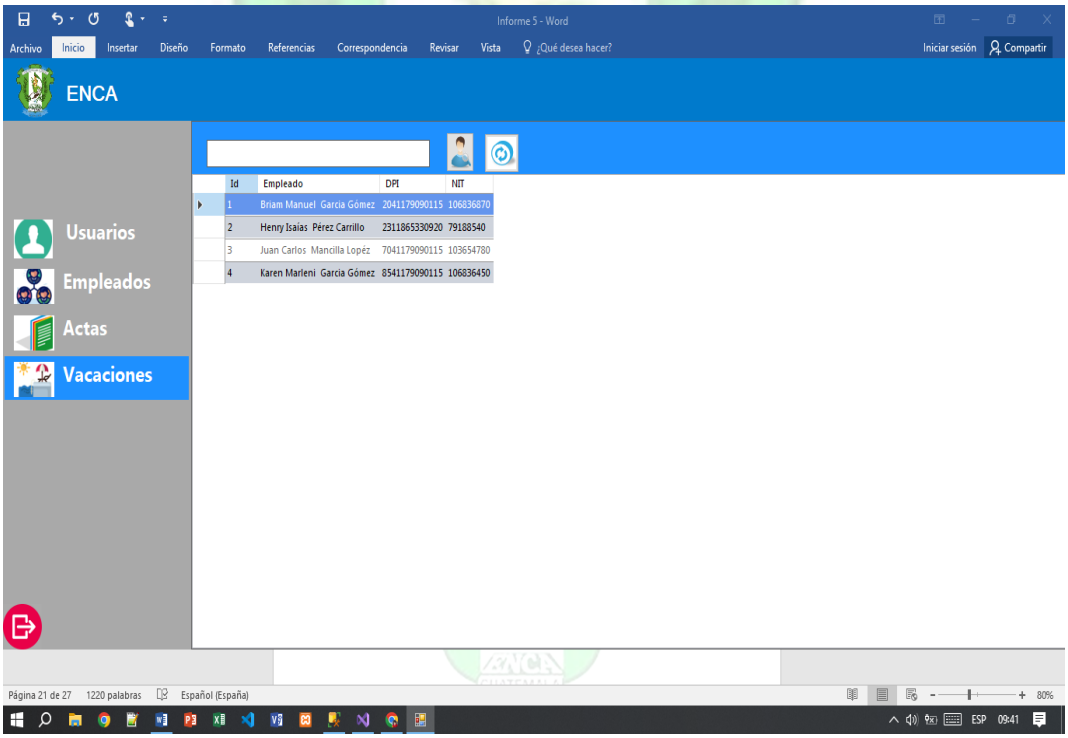
Esta es la funcionalidad de actas y el diseño de usuario.

Menú de Vacaciones.

La vacación está relacionada con el menú de empleados porque al ingresar un registro con el empleado se ingresa en la lista de empleados en el menú de empleados y de vacaciones.



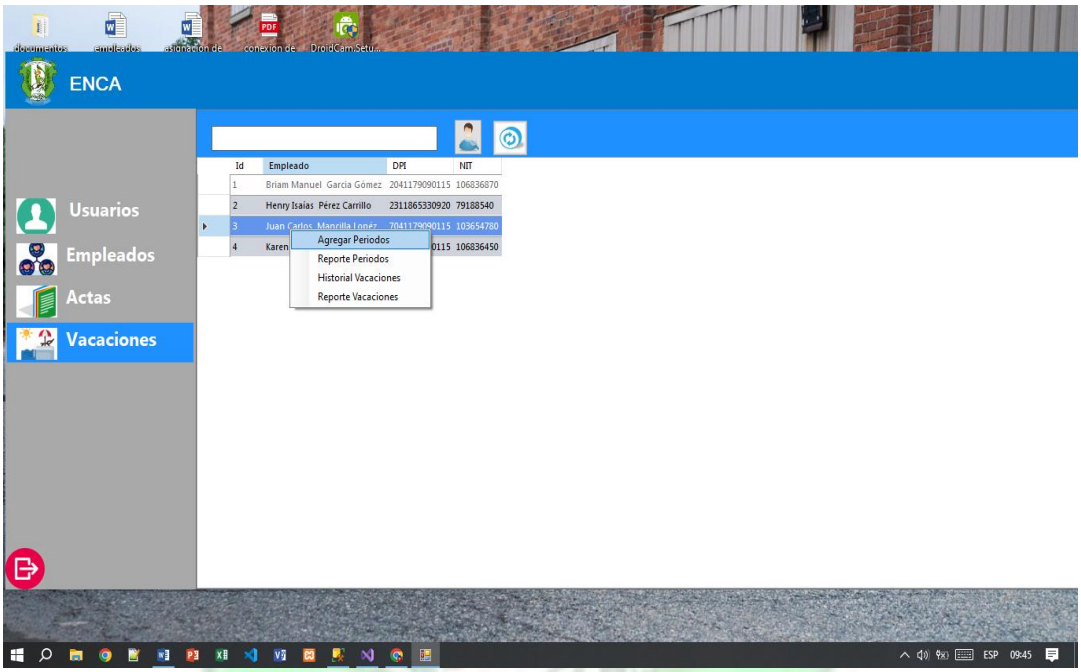
Como podemos ver,



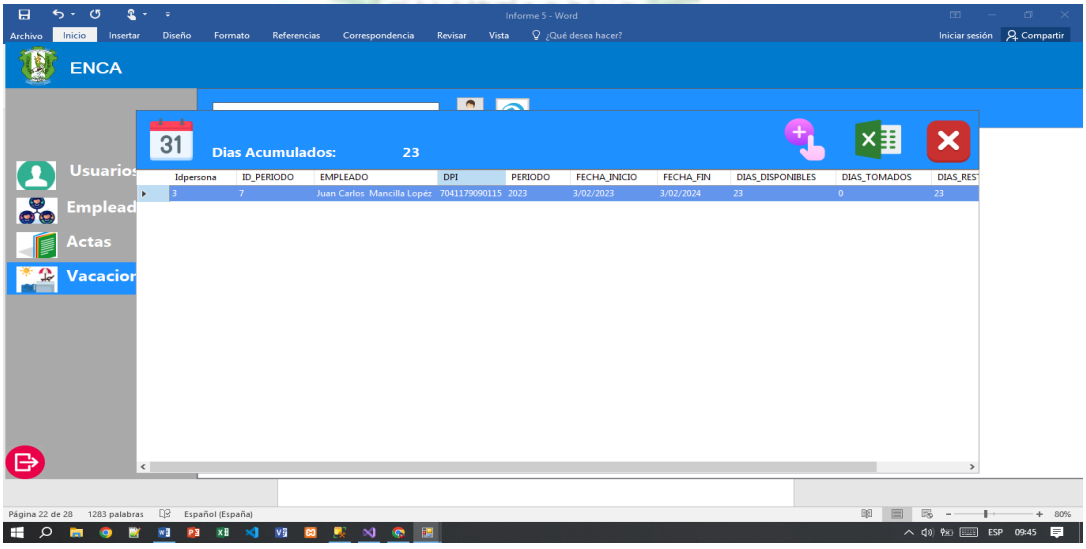
Solo que en empleados solo se tiene datos relevantes del empleado como su nombre, DPI y NIT.

Aquí en vacaciones tenemos el registro del empleado con sus datos más importante la funcionalidad de este interfaz es la siguiente.

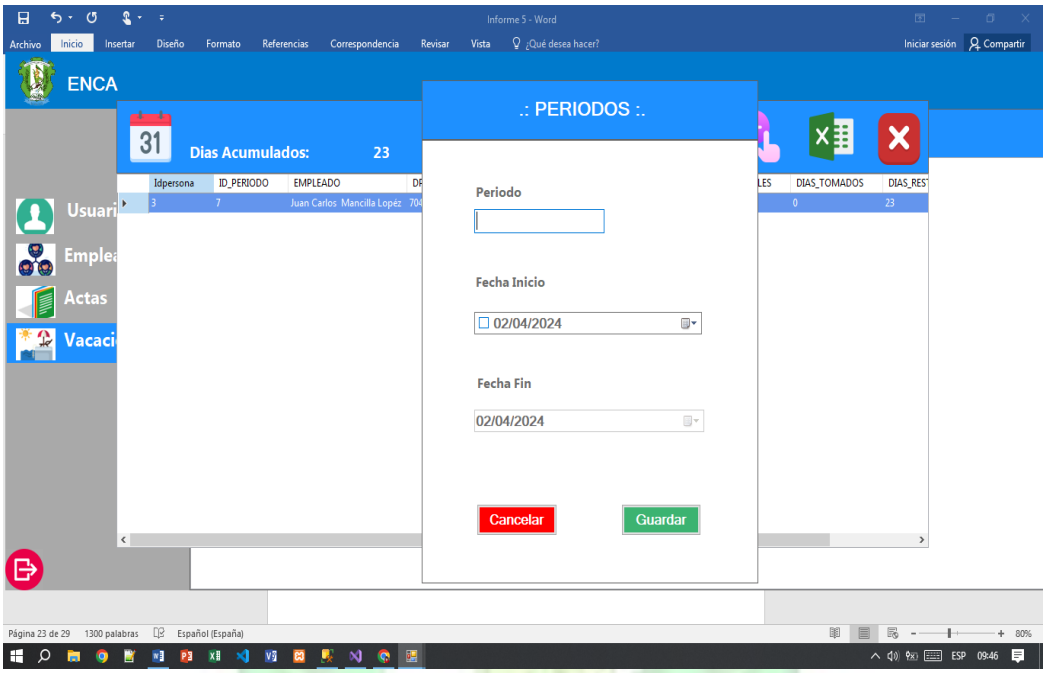
Al dar clic derecho sobre el registro de un empleado, aparece un sub menú el cual puede ingresar periodos de vacaciones al empleado.



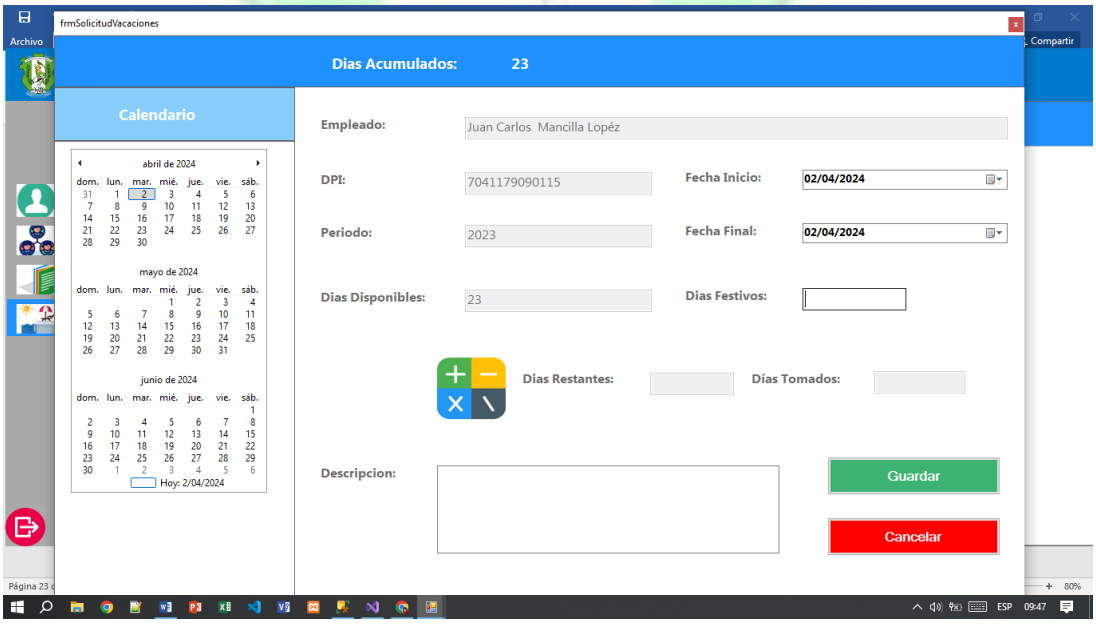
Damos clic.



Y nos muestra un registro de periodos del empleado, y también se puede ingresar un nuevo periodo.



Y en base a este periodo se puede hacer una asignación de vacaciones del empleado.



Creación de Módulo de Usuarios.

Creación de tablas usuario y tipo de Usuario y procedimientos almacenados.

Creación de tabla Usuario código.

```
USE [Enca]
GO

/****** Object:  Table [dbo].[Usuario]
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Usuario](
	[UsuarioID] [int] IDENTITY(1,1) NOT NULL,
	[NombreUsuario] [nvarchar](50) NOT NULL,
	[Contrasena] [nvarchar](50) NOT NULL,
	[Activo] [bit] NOT NULL,
	[TipoUsuarioID] [int] NULL,
	[Foto] [image] NULL,
PRIMARY KEY CLUSTERED
(
	[UsuarioID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

ALTER TABLE [dbo].[Usuario]  WITH CHECK ADD FOREIGN KEY([TipoUsuarioID])
REFERENCES [dbo].[TiposUsuario] ([TipoUsuarioID])
	GO
```

Creación de tabla tipo de usuario código.

```
USE [Enca]
GO

/****** Object:  Table [dbo].[TiposUsuario******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[TiposUsuario](
	[TipoUsuarioID] [int] IDENTITY(1,1) NOT NULL,
	[TipoUsuario] [nvarchar](50) NOT NULL,
PRIMARY KEY CLUSTERED
(
	[TipoUsuarioID] ASC
```

```
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY]
GO
```

Procedimientos almacenas código.

```
USE [Enca]
GO
/****** Object:  StoredProcedure [dbo].[sp_InsertarUsuario]    SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
Create PROCEDURE [dbo].[sp_InsertarUsuario]
    @NombreUsuario VARCHAR(50),
    @Contrasena VARCHAR(50),
    @Activo BIT,
    @TipoUsuario INT,
    @Foto IMAGE
AS
BEGIN
    INSERT INTO Usuario (NombreUsuario, Contrasena, Activo, TipoUsuarioID, Foto)
    VALUES (@NombreUsuario, @Contrasena, @Activo, @TipoUsuario, @Foto);
       END;
```

```
USE [Enca]
GO
/****** Object:  StoredProcedure [dbo].[sp_ListarTipoUsuario]    SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
create procedure [dbo].[sp_ListarTipoUsuario]
as
begin
select * from TiposUsuario
end;
```

```
USE [Enca]
GO
/****** Object:  StoredProcedure [dbo].[sp_ListarUsuarios]
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER procedure [dbo].[sp_ListarUsuarios]
as
begin;
select U.UsuarioID,
       U.NombreUsuario,
          U.Contrasena,
```

```
          T.TipoUsuario,
          U.Activo

from Usuario U
inner join TiposUsuario T on T.TipoUsuarioID = U.TipoUsuarioID
       end;
```

Programación de módulo de usuario en capas del sistema.

Capa de datos Código.

```
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 cUsuario
   {
      public int UsuarioID { get; set; }
      public string NombreUsuario { get; set; }
      public string Contrasena { get; set; }
      public bool Activo { get; set; }
      public int TipoUsuario { get; set; }
      public byte[] Foto { get; set; }


      public string BuscarUsuario {  get; set; }
```

```
    public string BuscarPersona { get; set; }
    public cUsuario()
    {
        // Constructor vacío
    }

    public cUsuario(int usuarioID, string nombreUsuario, string contrasena, bool activo, int tipoUsuario,
byte[] foto, string buscar, string buscarPersona)
    {
        UsuarioID = usuarioID;
        NombreUsuario = nombreUsuario;
        Contrasena = contrasena;
        Activo = activo;
        TipoUsuario = tipoUsuario;
        Foto = foto;
        BuscarUsuario = buscar;
        BuscarPersona = buscarPersona;
    }
    public DataTable Mostrar()
    {
        DataTable DtResultado = new DataTable("Usuario");
        SqlConnection sqlconn = new SqlConnection();
        try
        {
            sqlconn.ConnectionString = Conexion.Cn;
            SqlCommand sqlcmd = new SqlCommand();
            sqlcmd.Connection = sqlconn;
            sqlcmd.CommandText = "sp_ListarUsuarios";
            sqlcmd.CommandType = CommandType.StoredProcedure;
```

```
        SqlDataAdapter sqlDat = new SqlDataAdapter(sqlcmd);
        sqlDat.Fill(DtResultado);
    }
    catch (Exception ex)
    {
        DtResultado = null;
    }
    return DtResultado;
}


public DataTable MostrarPersonaEstado()
{
    DataTable DtResultado = new DataTable("RHPersona");
    SqlConnection sqlconn = new SqlConnection();
    try
    {
        sqlconn.ConnectionString = Conexion.Cn;
        SqlCommand sqlcmd = new SqlCommand();
        sqlcmd.Connection = sqlconn;
        sqlcmd.CommandText = "sp_MostrarRHPersonaestado";
        sqlcmd.CommandType = CommandType.StoredProcedure;

        SqlDataAdapter sqlDat = new SqlDataAdapter(sqlcmd);
        sqlDat.Fill(DtResultado);
    }
    catch (Exception ex)
    {
```

```
        DtResultado = null;
    }
    return DtResultado;
}

public DataTable MostrarActasEstado()
{
    DataTable DtResultado = new DataTable("RHACTAPOSESION");
    SqlConnection sqlconn = new SqlConnection();
    try
    {
        sqlconn.ConnectionString = Conexion.Cn;
        SqlCommand sqlcmd = new SqlCommand();
        sqlcmd.Connection = sqlconn;
        sqlcmd.CommandText = "sp_ListarRHActaMantenimiento";
        sqlcmd.CommandType = CommandType.StoredProcedure;

        SqlDataAdapter sqlDat = new SqlDataAdapter(sqlcmd);
        sqlDat.Fill(DtResultado);
    }
    catch (Exception ex)
    {
        DtResultado = null;
    }
    return DtResultado;
}

public string ActualizarEstadoPersona(int idPeriodo, bool estado)
{
```

```
    string rpta = "";

    using (SqlConnection sqlcon = new SqlConnection(Conexion.Cn))
    {
        try
        {
            sqlcon.Open();

            using (SqlCommand cmd = new SqlCommand("sp_ActualizarEstadoPersona", sqlcon))
            {
                cmd.CommandType = CommandType.StoredProcedure;

                cmd.Parameters.AddWithValue("@ID", idPeriodo).SqlDbType = SqlDbType.Int;
                cmd.Parameters.AddWithValue("@Estado", estado).SqlDbType = SqlDbType.Bit;

                int filasAfectadas = cmd.ExecuteNonQuery();
                rpta = filasAfectadas > 0 ? "OK" : "No se actualizó ningún registro";
            }
        }
        catch (Exception ex)
        {
            rpta = ex.Message;
        }
    }

    return rpta;
}

public string ActualizarEstadoUsuario(int idPeriodo, bool estado)
```

```
{
    string rpta = "";

    using (SqlConnection sqlcon = new SqlConnection(Conexion.Cn))
    {
        try
        {
            sqlcon.Open();

            using (SqlCommand cmd = new SqlCommand("sp_ActualizarEstadoUsuario", sqlcon))
            {
                cmd.CommandType = CommandType.StoredProcedure;

                cmd.Parameters.AddWithValue("@Id", idPeriodo).SqlDbType = SqlDbType.Int;
                cmd.Parameters.AddWithValue("@Estado", estado).SqlDbType = SqlDbType.Bit;

                int filasAfectadas = cmd.ExecuteNonQuery();
                rpta = filasAfectadas > 0 ? "OK" : "No se actualizó ningún registro";
            }
        }
        catch (Exception ex)
        {
            rpta = ex.Message;
        }
    }

    return rpta;
}
public string ActualizarEstadoActa(int idActa)
```

```
{
    string rpta = "";

    using (SqlConnection sqlcon = new SqlConnection(Conexion.Cn))
    {
        try
        {
            sqlcon.Open();

            using (SqlCommand cmd = new SqlCommand("sp_DesactivarActa", sqlcon))
            {
                cmd.CommandType = CommandType.StoredProcedure;

                cmd.Parameters.AddWithValue("@Id", idActa).SqlDbType = SqlDbType.Int;

                int filasAfectadas = cmd.ExecuteNonQuery();
                rpta = filasAfectadas > 0 ? "OK" : "No se actualizó ningún registro";
            }
        }
        catch (Exception ex)
        {
            rpta = ex.Message;
        }
    }

    return rpta;
}
```

```
public string Insertar(cUsuario usuario)
{
   string respuesta = "";
   using (SqlConnection connection = new SqlConnection(Conexion.Cn))
   {
      try
      {
         connection.Open();
         using (SqlCommand cmd = new SqlCommand("sp_InsertarUsuario", connection))
         {
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Parameters.AddWithValue("@NombreUsuario", usuario.NombreUsuario);
            cmd.Parameters.AddWithValue("@Contrasena", usuario.Contrasena);
            cmd.Parameters.AddWithValue("@Activo", usuario.Activo);
            cmd.Parameters.AddWithValue("@TipoUsuario", usuario.TipoUsuario);
            cmd.Parameters.AddWithValue("@Foto", usuario.Foto);

            respuesta = cmd.ExecuteNonQuery() == 1 ? "OK" : "Error al insertar usuario";
         }
      }
      catch (Exception ex)
      {
         respuesta = ex.Message;
      }
   }
   return respuesta;
}

public string Actualizar(cUsuario usuario)
```

```
{
    string respuesta = "";
    using (SqlConnection connection = new SqlConnection(Conexion.Cn))
    {
        try
        {
            connection.Open();
            using (SqlCommand cmd = new SqlCommand("sp_ActualizarUsuario", connection))
            {
                cmd.CommandType = CommandType.StoredProcedure;
                cmd.Parameters.AddWithValue("@UsuarioID", usuario.UsuarioID);
                cmd.Parameters.AddWithValue("@NombreUsuario", usuario.NombreUsuario);
                cmd.Parameters.AddWithValue("@Contrasena", usuario.Contrasena);
                cmd.Parameters.AddWithValue("@Activo", usuario.Activo);
                cmd.Parameters.AddWithValue("@TipoUsuario", usuario.TipoUsuario);
                cmd.Parameters.AddWithValue("@Foto", usuario.Foto);

                respuesta = cmd.ExecuteNonQuery() == 1 ? "OK" : "Error al actualizar usuario";
            }
        }
        catch (Exception ex)
        {
            respuesta = ex.Message;
        }
    }
    return respuesta;
}

public DataTable Buscar(cUsuario usuario)
```

```
{
    DataTable DtResultado1 = new DataTable("Usuario");
    SqlConnection sqlconn = new SqlConnection();
    try
    {
        sqlconn.ConnectionString = Conexion.Cn;
        SqlCommand sqlcmd = new SqlCommand();
        sqlcmd.Connection = sqlconn;
        sqlcmd.CommandText = "sp_BuscarUsuarioPorNombre";
        sqlcmd.CommandType = CommandType.StoredProcedure;

        SqlParameter ParBuscar = new SqlParameter();
        ParBuscar.ParameterName = "@NombreUsuario";
        ParBuscar.SqlDbType = SqlDbType.VarChar;
        ParBuscar.Size = 50;
        ParBuscar.Value = usuario.BuscarUsuario;
        sqlcmd.Parameters.Add(ParBuscar);

        SqlDataAdapter sqlDat = new SqlDataAdapter(sqlcmd);
        sqlDat.Fill(DtResultado1);
    }
    catch (Exception ex)
    {
        DtResultado1 = null;
    }
    return DtResultado1;
}

public DataTable BuscarPersonaId(cUsuario usuario)
```

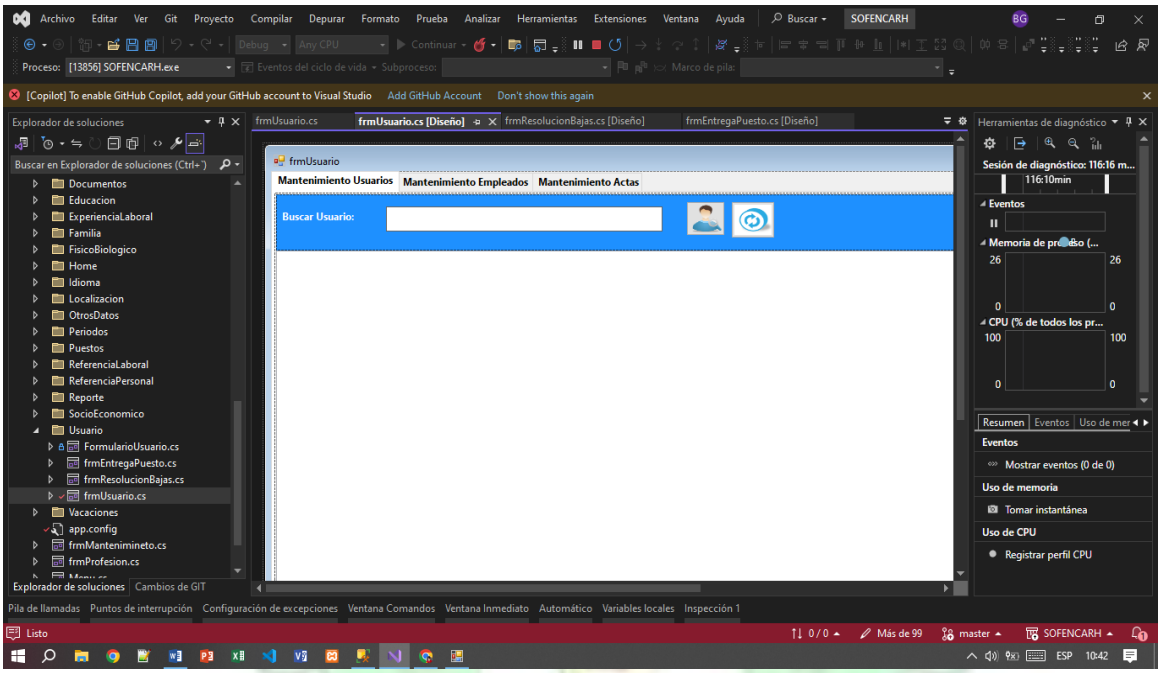
```
    {
        DataTable DtResultado1 = new DataTable("RHPersona");
        SqlConnection sqlconn = new SqlConnection();
        try
        {
            sqlconn.ConnectionString = Conexion.Cn;
            SqlCommand sqlcmd = new SqlCommand();
            sqlcmd.Connection = sqlconn;
            sqlcmd.CommandText = "sp_MostrarRHPersonaestadoID";
            sqlcmd.CommandType = CommandType.StoredProcedure;

            SqlParameter ParBuscar = new SqlParameter();
            ParBuscar.ParameterName = "@Persona";
            ParBuscar.SqlDbType = SqlDbType.VarChar;
            ParBuscar.Size = 200;
            ParBuscar.Value = usuario.BuscarPersona;
            sqlcmd.Parameters.Add(ParBuscar);

            SqlDataAdapter sqlDat = new SqlDataAdapter(sqlcmd);
            sqlDat.Fill(DtResultado1);
        }
        catch (Exception ex)
        {
            DtResultado1 = null;
        }
        return DtResultado1;
    }
  }
}
```

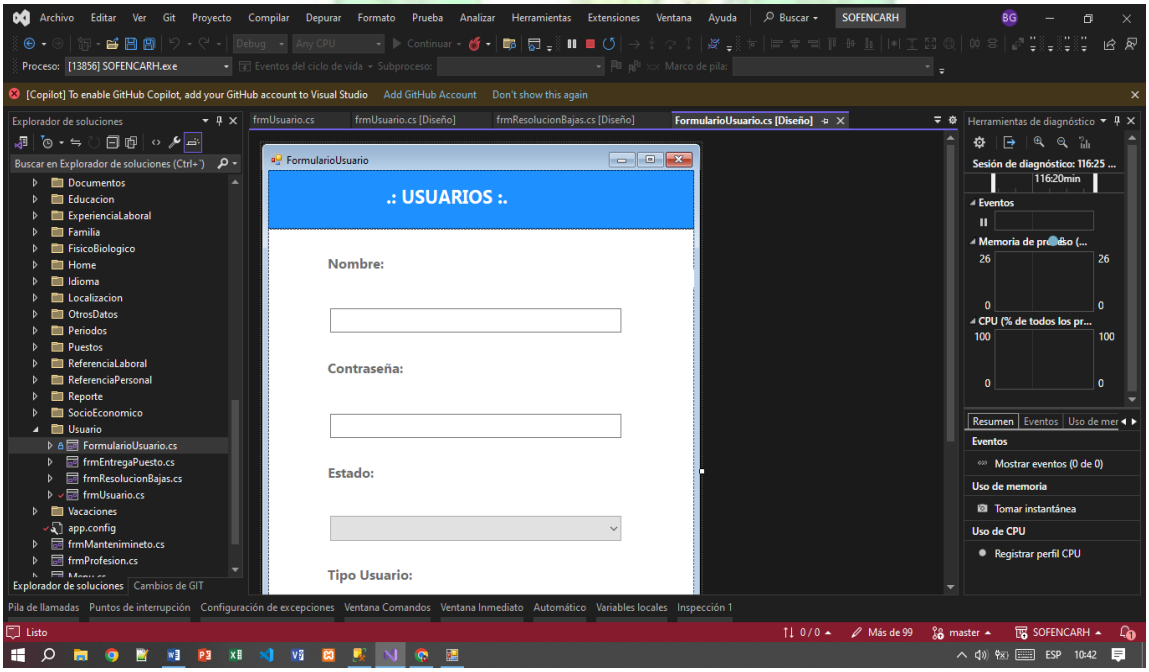
Capa de negocio.

```
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 nUsuarios
    {


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

        public static DataTable MostrarPersonaEstado()
        {
            return new cUsuario().MostrarPersonaEstado();
        }
        public static DataTable MostrarActasEstado()
        {
            return new cUsuario().MostrarActasEstado();
        }
        public static string ActualizarEstadoPersona(int idPeriodo, bool estado)
        {
            cUsuario Obj = new cUsuario();
            return Obj.ActualizarEstadoPersona(idPeriodo, estado);

        }

        public static string ActualizarEstadoUsuario(int idPeriodo, bool estado)
        {
            cUsuario Obj = new cUsuario();
            return Obj.ActualizarEstadoUsuario(idPeriodo, estado);

        }

        public static string DesactivarActa(int idPeriodo)
        {
            cUsuario Obj = new cUsuario();
            return Obj.ActualizarEstadoActa(idPeriodo);

        }
        public static string InsertarUsuario( string nombreUsuario, string
contrasena, bool activo, int tipoUsuario, byte[] foto)
        {
            // Crear una instancia de la clase Usuario (suponiendo que ya tienes
una)
            cUsuario usuario = new cUsuario();

            // Establecer las propiedades del objeto con los valores proporcionados
```

```
            usuario.NombreUsuario = nombreUsuario;
            usuario.Contrasena = contrasena;
            usuario.Activo = activo;
            usuario.TipoUsuario = tipoUsuario;
            usuario.Foto = foto;

            // Llamar al método de actualización en la clase Usuario (debes tener
este método)
            return usuario.Insertar(usuario);
        }
        public static string EditarUsuario(int id, string nombreUsuario, string
contrasena, bool activo, int tipoUsuario,byte[]foto)
        {
            // Crear una instancia de la clase Usuario (suponiendo que ya tienes
una)
            cUsuario usuario = new cUsuario();

            // Establecer las propiedades del objeto con los valores proporcionados
            usuario.UsuarioID = id;
            usuario.NombreUsuario = nombreUsuario;
            usuario.Contrasena = contrasena;
            usuario.Activo = activo;
            usuario.TipoUsuario = tipoUsuario;
            usuario.Foto = foto;

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

        public static DataTable BuscarUsuario(string textoBuscar)
        {
            cUsuario usuario = new cUsuario();
            usuario.BuscarUsuario = textoBuscar;
            return usuario.Buscar(usuario);
        }

        public static DataTable BuscarPersona(string textoBuscar)
        {
            cUsuario usuario = new cUsuario();
            usuario.BuscarPersona = textoBuscar;
            return usuario.BuscarPersonaId(usuario);
        }


    }
}
```

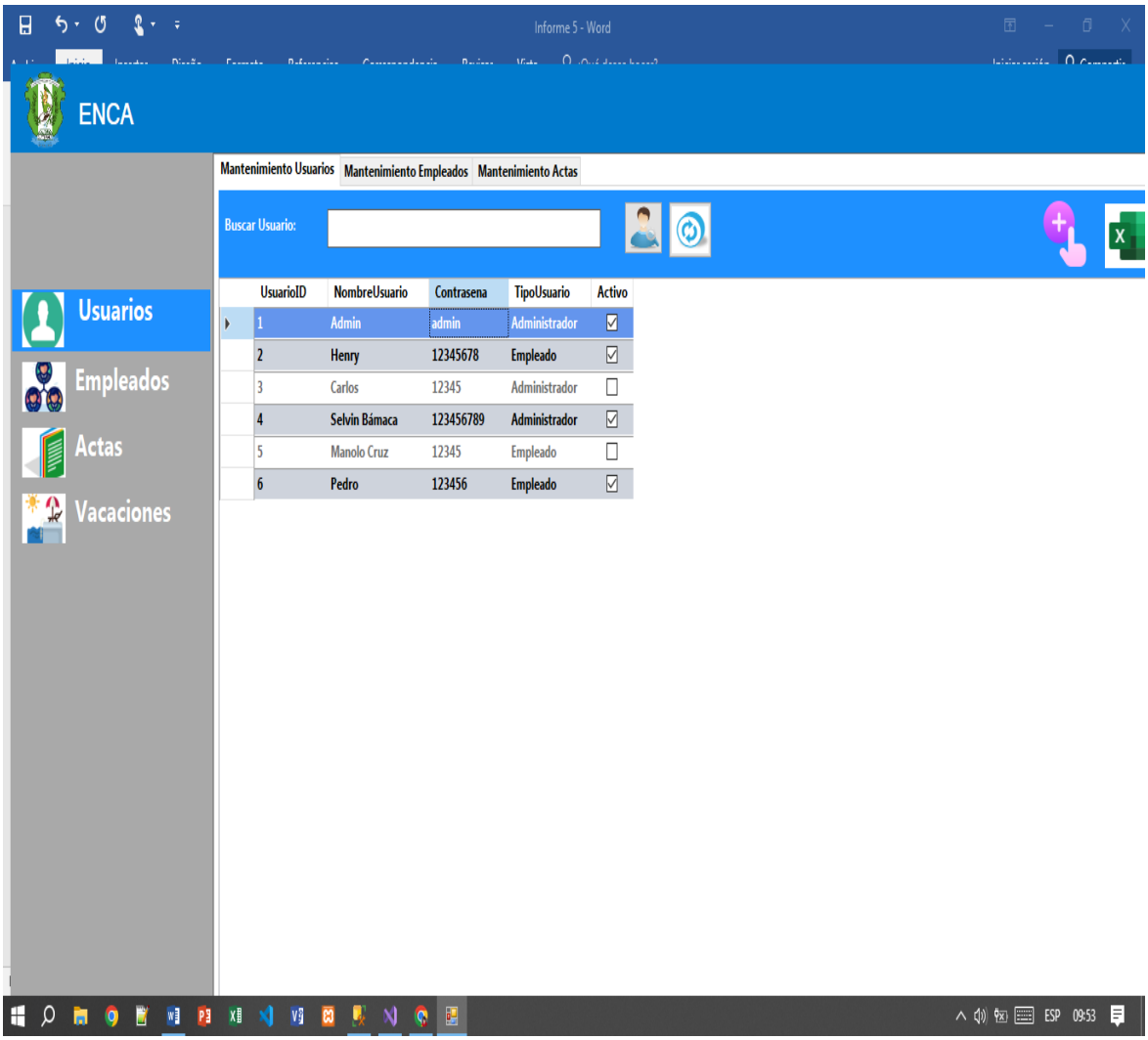
Capa vista.


frmUsuario
Mantenimiento Usuarios
Mantenimiento Empleados
Mantenimiento Actas
Buscar Usuario:

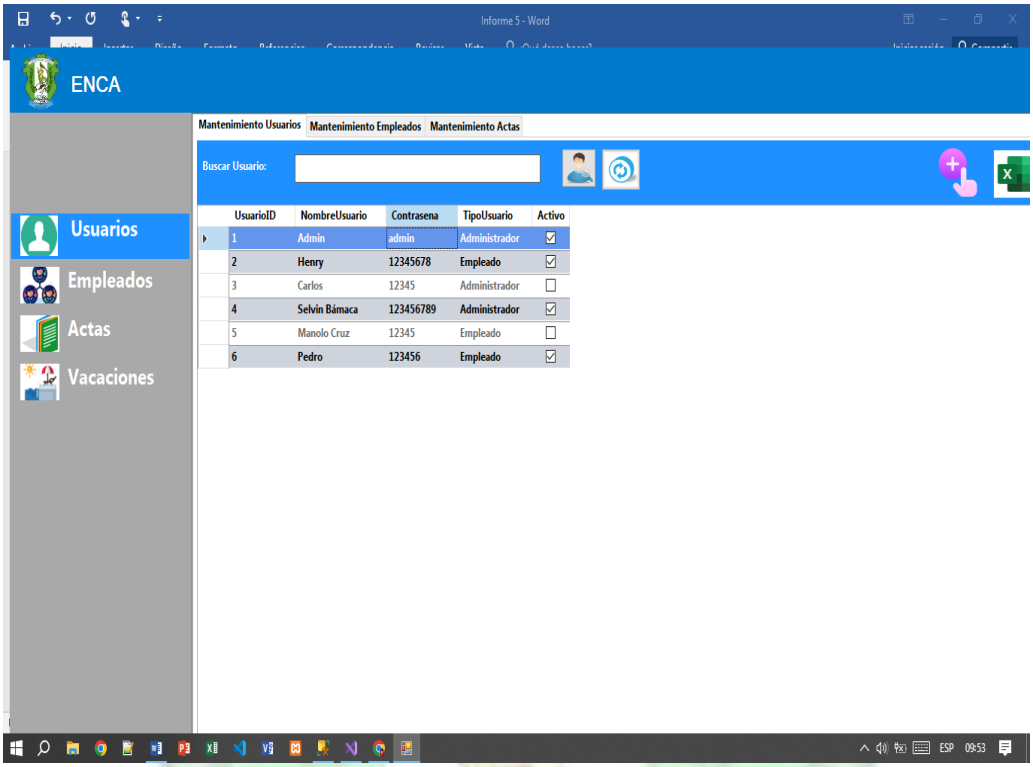

FormularioUsuario
.: USUARIOS :.
Nombre:
Contraseña:
Estado:
Tipo Usuario:

Interfaz de Modulo de usuario.

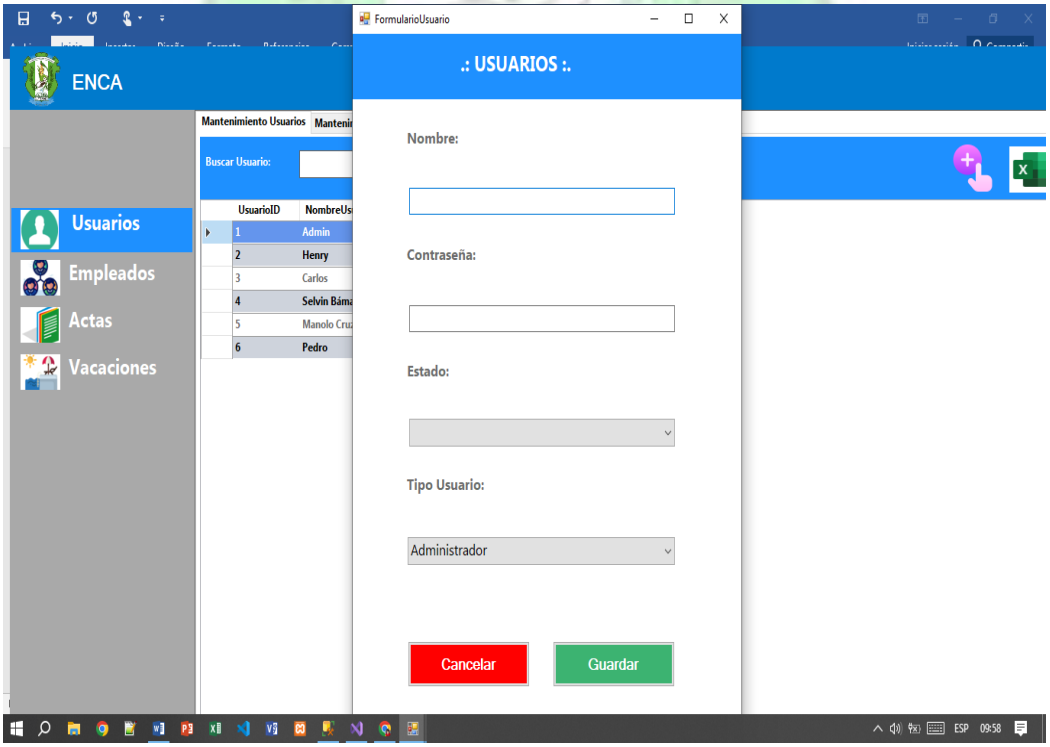
En base a la necesidad de este módulo es para que un usuario administrador del sistema pueda tener los privilegios maestros del sistema el cual va crear usuarios de todo tipo para ingresar al sistema y con ciertos privilegios.



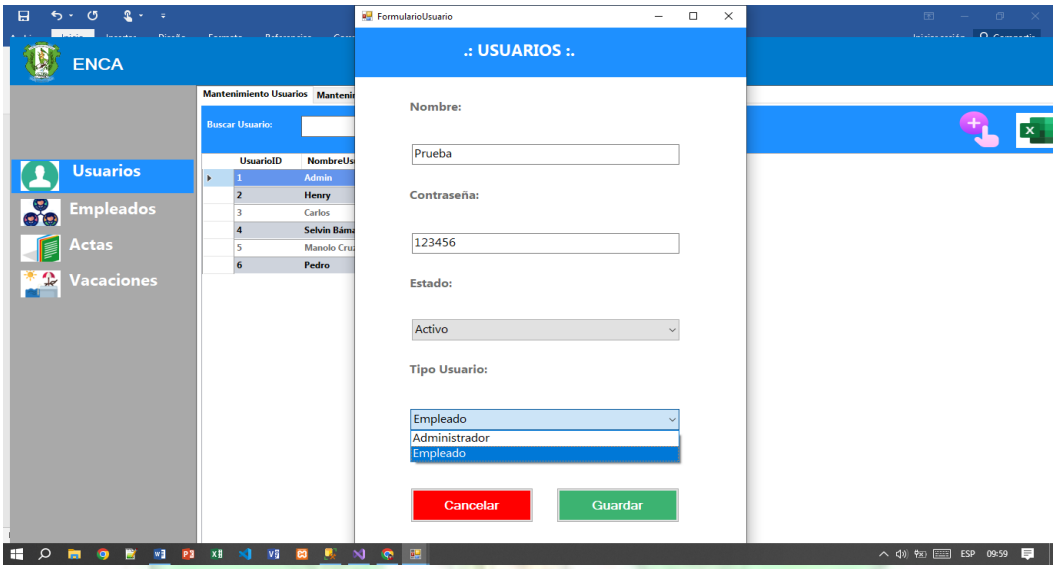
El usuario administrador debe de crear usuarios. Para ello debe de seguir el siguiente procedimiento.



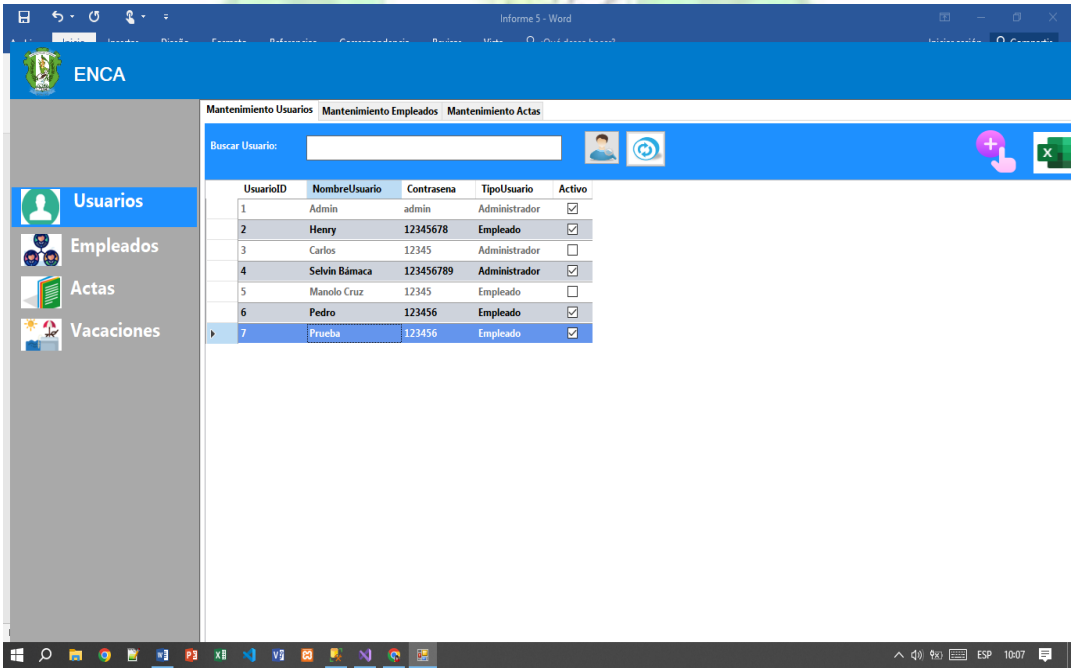
Aquí donde está el botón de la manita con el signo de más.



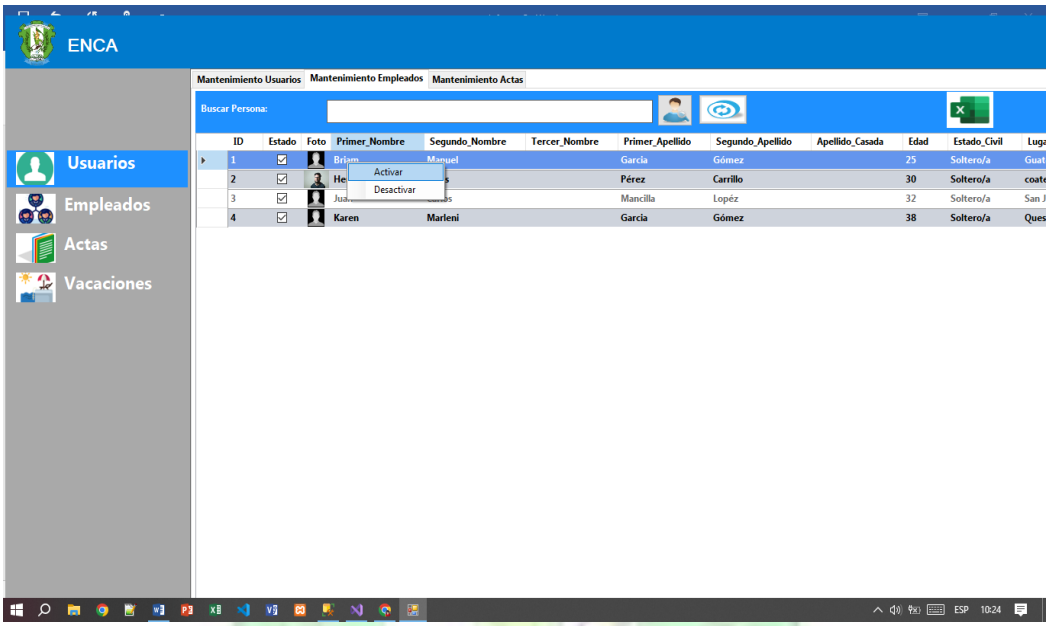
Nos nuestra la pantalla para poder ingresar un nuevo usuario para administrar el sistema con ciertos privilegios.

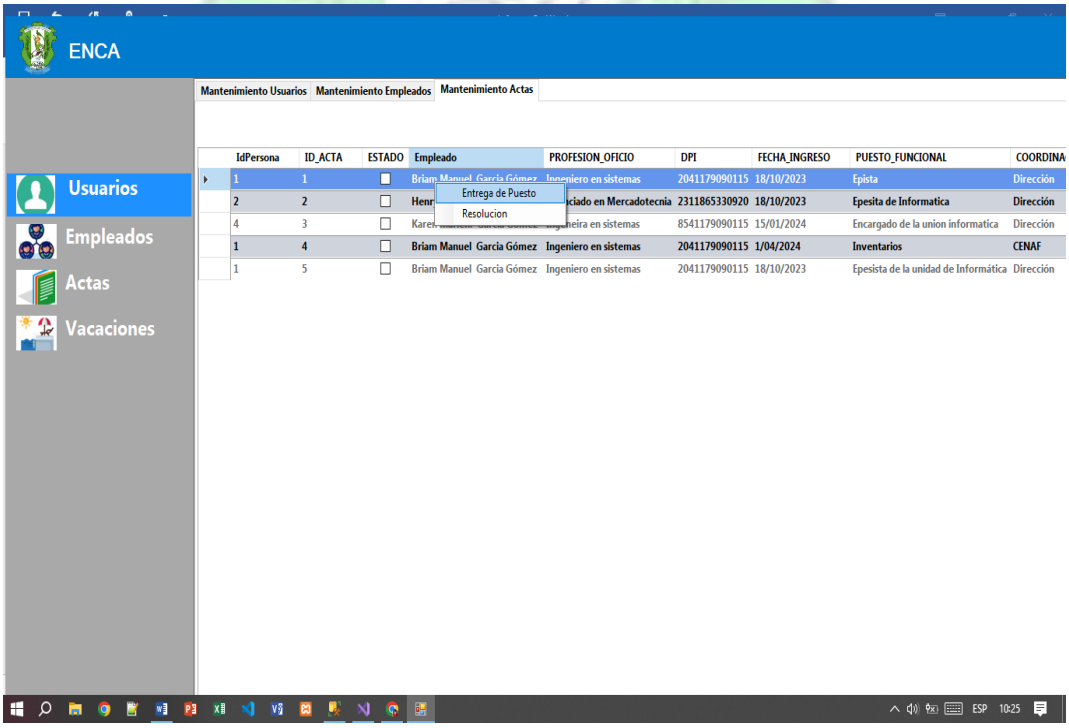


Aquí definimos sus privilegios del sistema.



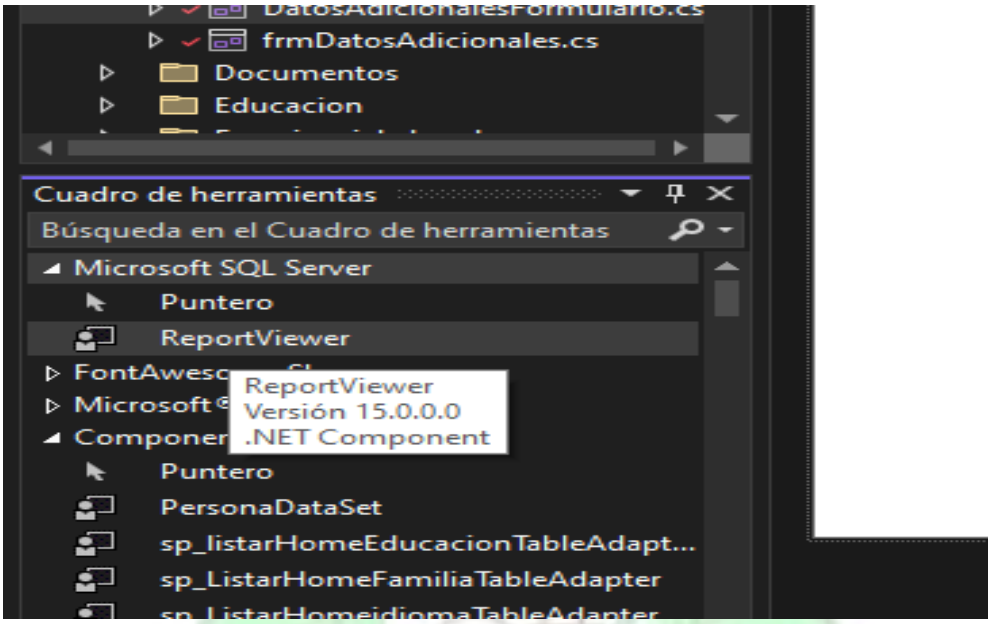
Aquí ya este nuevo usuario tiene acceso al sistema, esta es una de las funcionalidades de este módulo también desde este se puede desactivar los empleados y toda su información y también las actas.





Creación de Reportes.

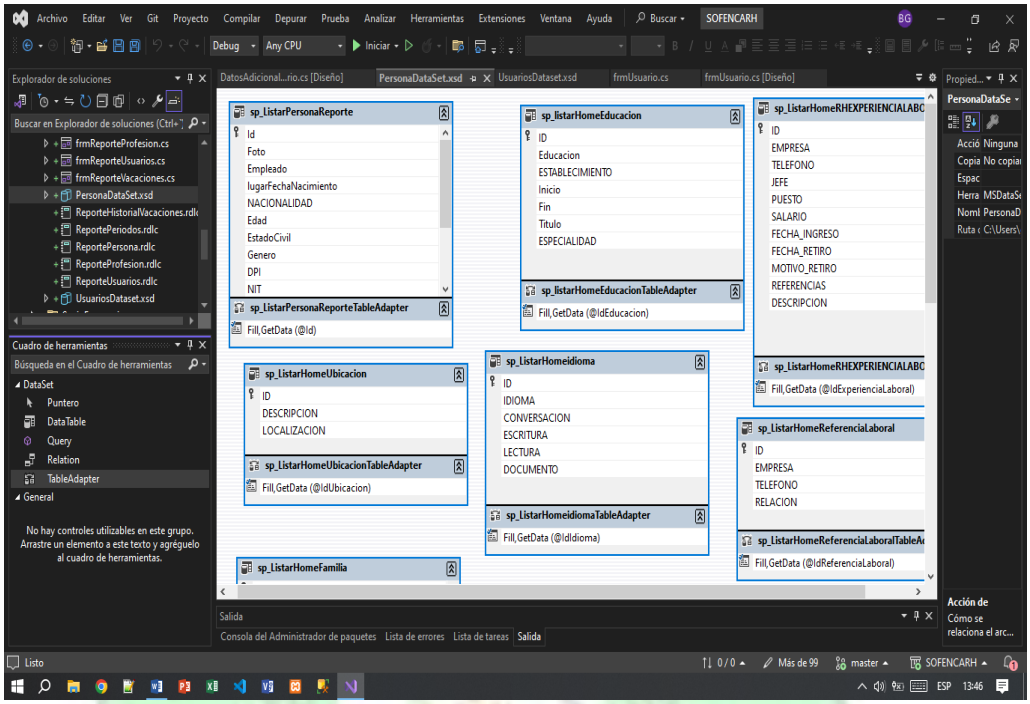
Para la creación de todos los reportes se está utilizando el creador de reportes de Windows Report viwer.

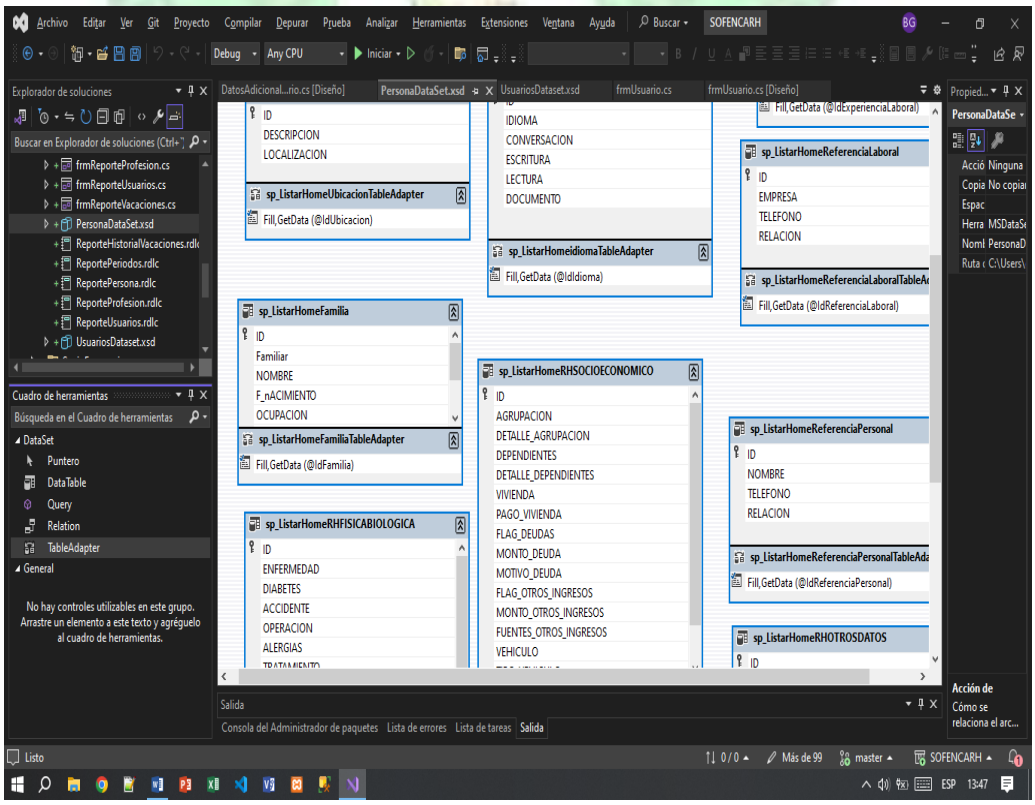


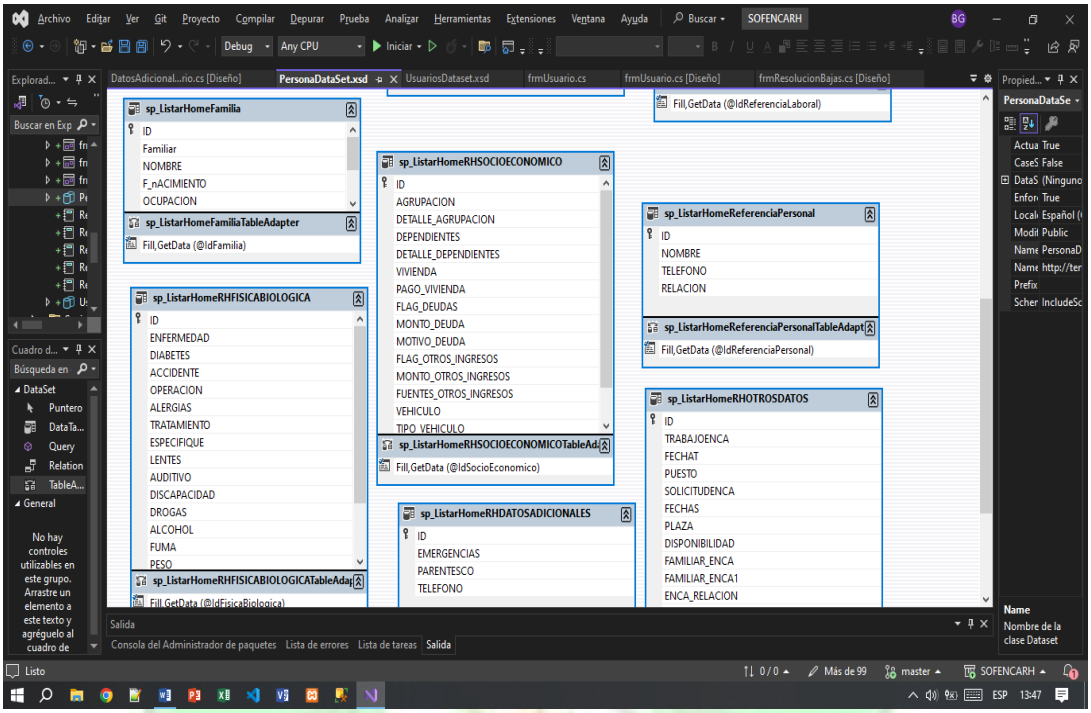
Este reportador me permite hacer reportes en base a la lógica de mi base de datos la cual se maneja por los procedimientos almacenados que se programaron.

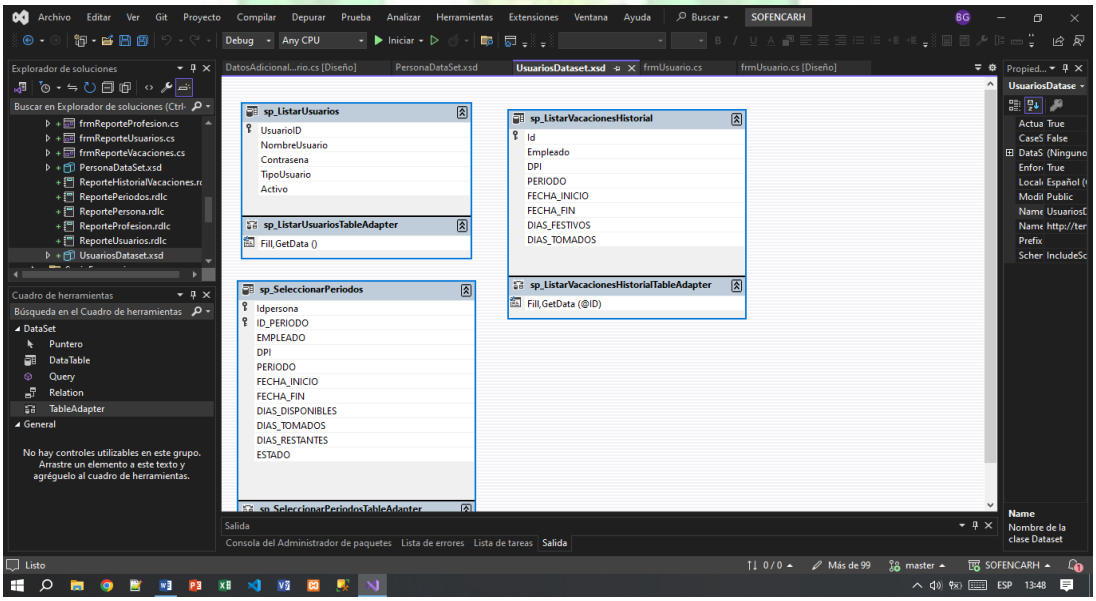
Para realizar los reportes se deben de realizar los siguientes pasos.

1. En nuestra capa de vista se crea una nueva carpeta que contener los siguientes elementos, un dataset el cual va permitir hacer la conexión a nuestro servidor de base datos, para seleccionar los procedimientos almacenados que me permiten seleccionar los datos que se están ingresando a mi base de datos.


sp_ListarPersonaReporte
Id
Foto
Empleado
lugarFechaNacimiento
NACIONALIDAD
Edad
EstadoCivil
Genero
DPI
NIT
sp_ListarPersonaReporteTableAdapter
Fill,GetData (@Id)
sp_ListarHomeEducacion
ID
Educacion
ESTABLECIMIENTO
Inicio
Fin
Titulo
ESPECIALIDAD
sp_ListarHomeEducacionTableAdapter
Fill,GetData (@IdEducacion)
sp_ListarHomeRHEXPERIENCIALABC
ID
EMPRESA
TELEFONO
JEFE
PUESTO
SALARIO
FECHA_INGRESO
FECHA_RETIRO
MOTIVO_RETIRO
REFERENCIAS
DESCRIPCION
sp_ListarHomeRHEXPERIENCIALABC
Fill,GetData (@IdExperienciaLaboral)
sp_ListarHomeUbicacion
ID
DESCRIPCION
LOCALIZACION
sp_ListarHomeUbicacionTableAdapter
Fill,GetData (@IdUbicacion)
sp_ListarHomeIdioma
ID
IDIOMA
CONVERSACION
ESCRITURA
LECTURA
DOCUMENTO
sp_ListarHomeIdiomaTableAdapter
Fill,GetData (@Ididioma)
sp_ListarHomeReferenciaLaboral
ID
EMPRESA
TELEFONO
RELACION
sp_ListarHomeReferenciaLaboralTableA
Fill,GetData (@IdReferenciaLaboral)
sp_ListarHomeFamilia
Explorador de soluciones
Cuadro de herramientas
DataSet
Puntero
DataTable
Query
Relation
TableAdapter
General
No hay controles utilizables en este grupo. Arrastre un elemento a este texto y agréguelo al cuadro de herramientas.
Salida
Consola del Administrador de paquetes
Lista de errores
Lista de tareas
Salida
Listo
SOFENCARH
master
ESP
13:46


ID
DESCRIPCION
LOCALIZACION
sp_ListarHomeUbicacionTableAdapter
Fill,GetData (@IdUbicacion)
IDIOMA
CONVERSACION
ESCRITURA
LECTURA
DOCUMENTO
sp_ListarHomeidiomaTableAdapter
Fill,GetData (@Ididioma)
sp_ListarHomeReferenciaLaboral
ID
EMPRESA
TELEFONO
RELACION
sp_ListarHomeReferenciaLaboralTableA
Fill,GetData (@IdReferenciaLaboral)
sp_ListarHomeFamilia
ID
Familiar
NOMBRE
F_nACIMIENTO
OCUPACION
sp_ListarHomeFamiliaTableAdapter
Fill,GetData (@IdFamilia)
sp_ListarHomeRHSOCIOECONOMICO
ID
AGRUPACION
DETALLE_AGRUPACION
DEPENDIENTES
DETALLE_DEPENDIENTES
VIVIENDA
PAGO_VIVIENDA
FLAG_DEUDAS
MONTO_DEUDA
MOTIVO_DEUDA
FLAG_OTROS_INGRESOS
MONTO_OTROS_INGRESOS
FUENTES_OTROS_INGRESOS
VEHICULO
sp_ListarHomeReferenciaPersonal
ID
NOMBRE
TELEFONO
RELACION
sp_ListarHomeReferenciaPersonalTableAda
Fill,GetData (@IdReferenciaPersonal)
sp_ListarHomeRHFISICABIOLOGICA
ID
ENFERMEDAD
DIABETES
ACCIDENTE
OPERACION
ALERGIAS
sp_ListarHomeRHOTROSDATOS
ID
Explorador de soluciones
Cuadro de herramientas
DataSet
Puntero
DataTable
Query
Relation
TableAdapter
General
No hay controles utilizables en este grupo. Arrastre un elemento a este texto y agréguelo al cuadro de herramientas.
Salida
Consola del Administrador de paquetes
Lista de errores
Lista de tareas
Salida
Listo
SOFENCARH
master
ESP
13:47



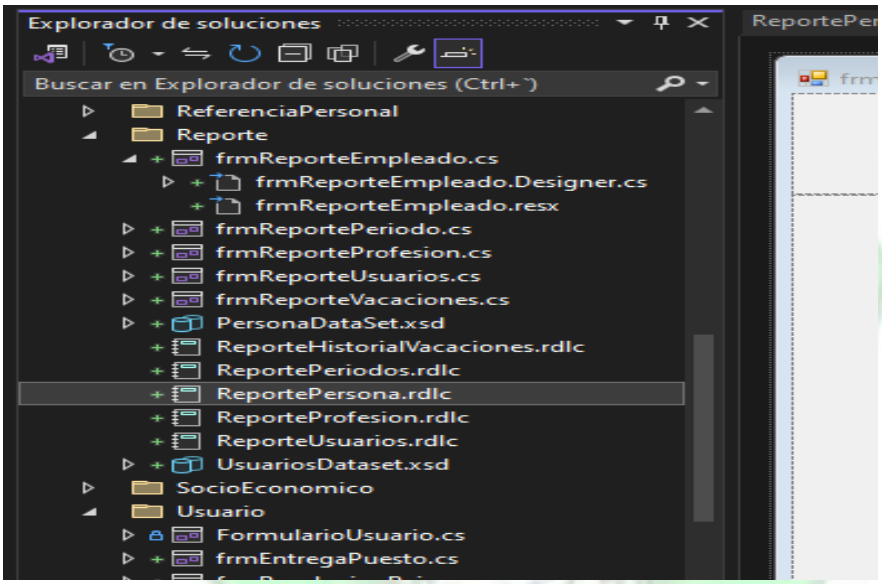


Estos dos dataset contienen los procedimientos almacenados de nuestras tablas cuando se ingresan la información estos traen esa información la cual nos sirve para poder crear nuestros reportes.

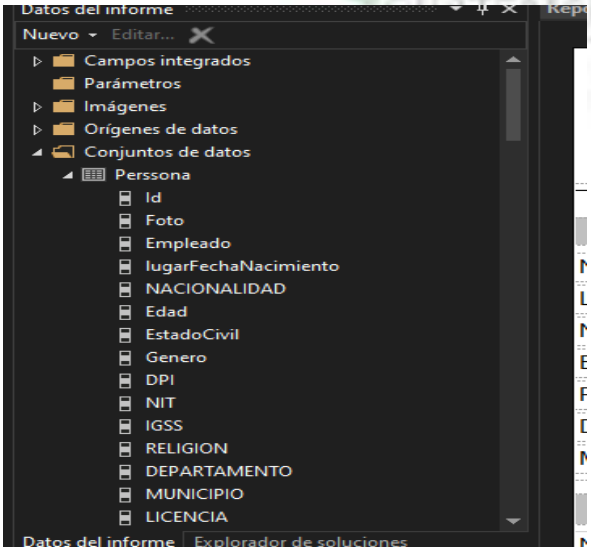
Para ello tenemos que crear nuestro reporte con un archivo de formato RDLC aquí vamos a definir nuestros reportes darles forma en base a los datos.

Reporte de empleado.

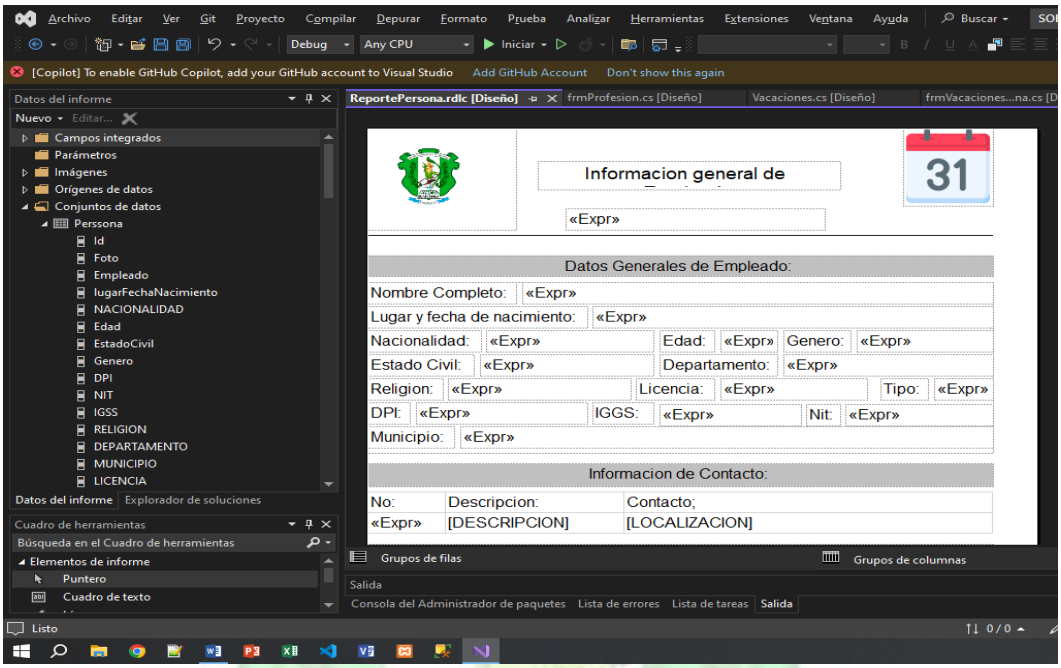
Primero se crea el reporte de persona el cual contiene toda la información del módulo de Perona,



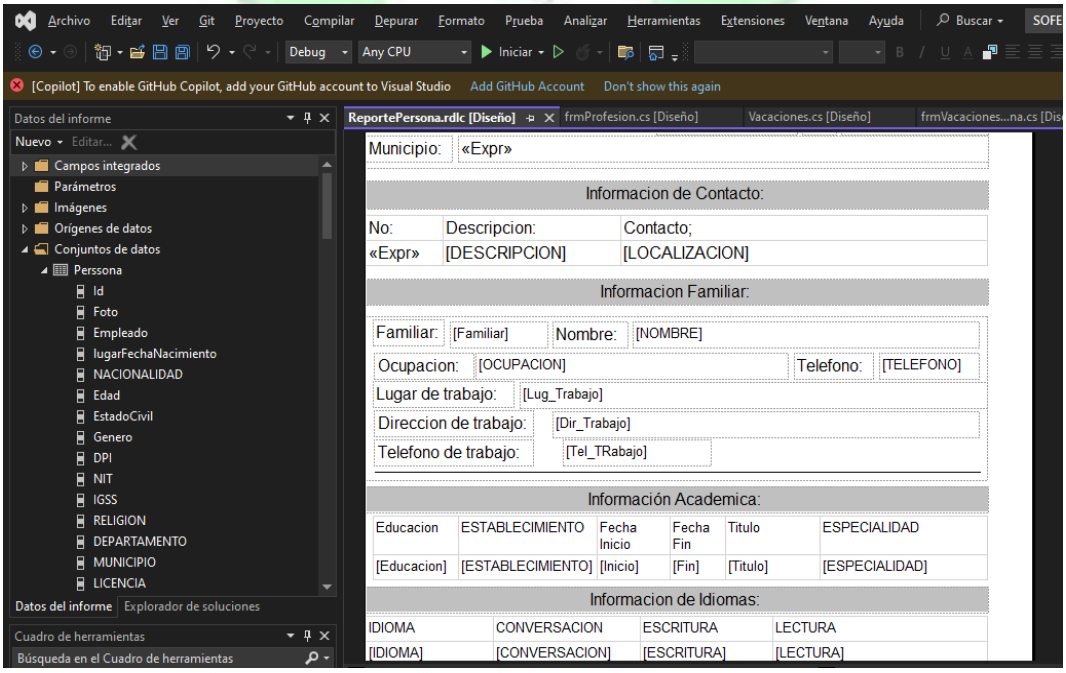
Aquí se hace el diseño y toda la lógica del reporte. Damos doble clic.

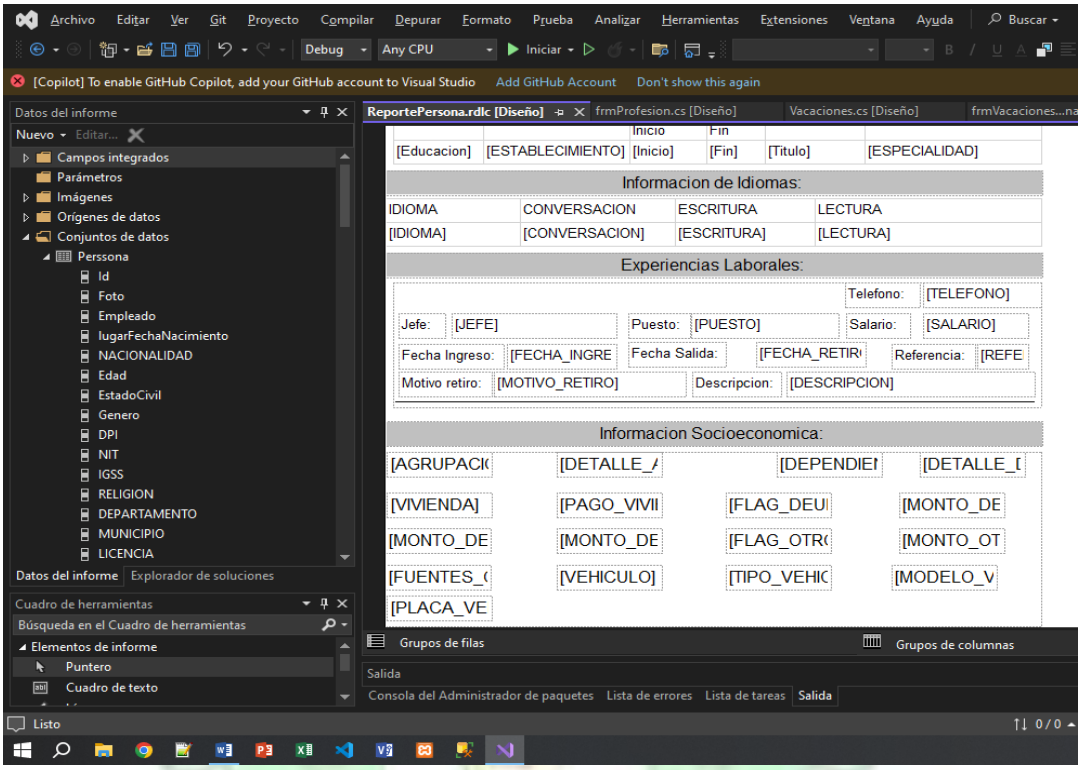


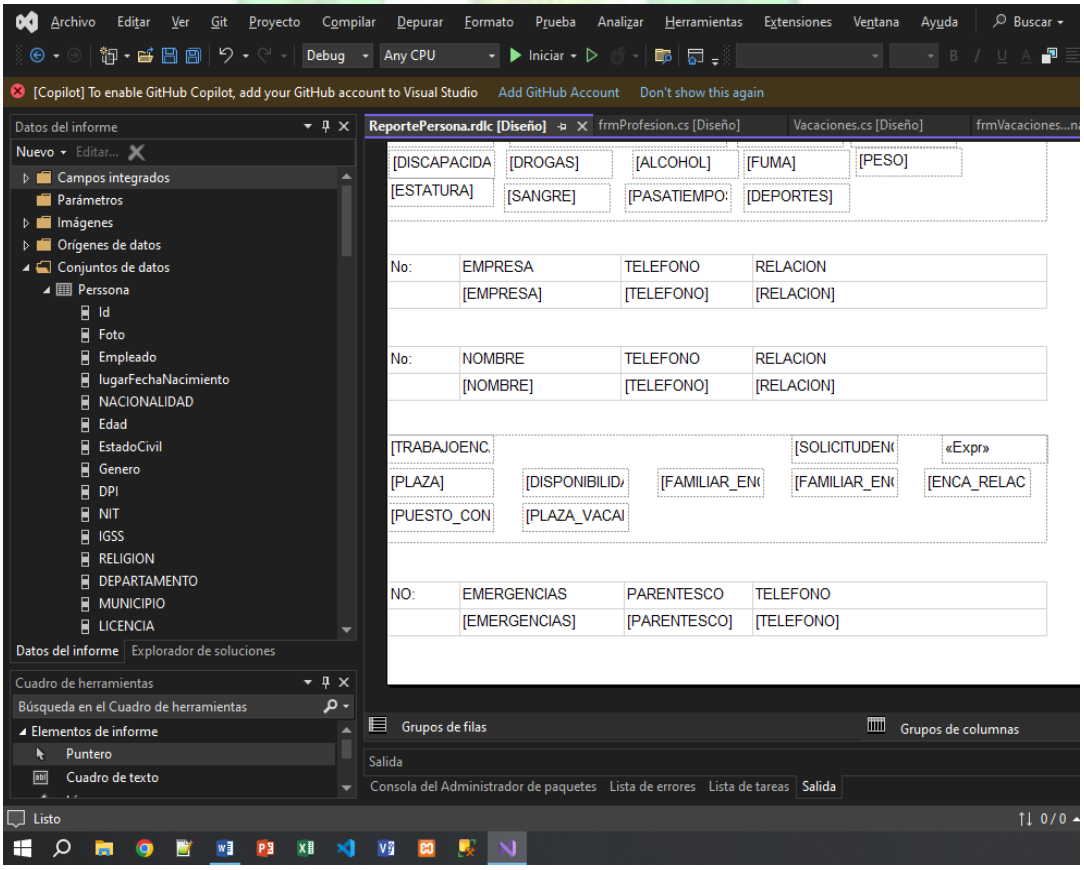
En conjunto de datos nos nuestras los elementos de los campos en base al procedimiento almacenado.



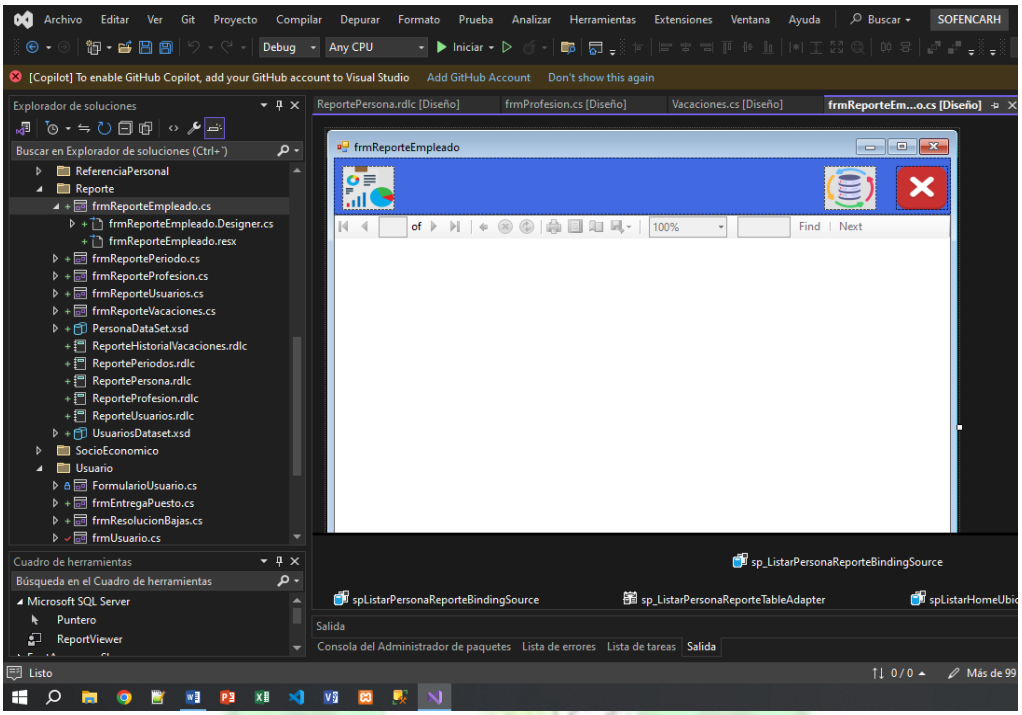
Y los arrastramos al reporte y así se van pegando y dando diseño a nuestro reporte en base a cada procedimiento almacenado que está en nuestro dataset.



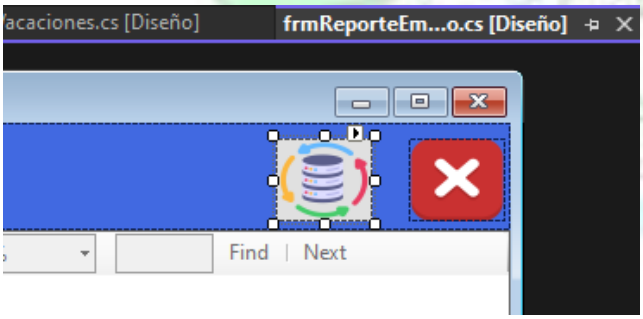

Archivo Editar Ver Git Proyecto Compilar Depurar Formato Prueba Analizar Herramientas Extensiones Ventana Ayuda Buscar
Debug Any CPU Iniciar
[Copilot] To enable GitHub Copilot, add your GitHub account to Visual Studio Add GitHub Account Don't show this again
Datos del informe
Nuevo Editar...
Campos integrados
Parámetros
Imágenes
Orígenes de datos
Conjuntos de datos
Perssona
Id
Foto
Empleado
lugarFechaNacimiento
NACIONALIDAD
Edad
EstadoCivil
Genero
DPI
NIT
IGSS
RELIGION
DEPARTAMENTO
MUNICIPIO
LICENCIA
Datos del informe Explorador de soluciones
Cuadro de herramientas
Búsqueda en el Cuadro de herramientas
Elementos de informe
Puntero
Cuadro de texto
ReportePersona.rdlc [Diseño] frmProfesion.cs [Diseño] Vacaciones.cs [Diseño] frmVacaciones...
Inicio Fin
[Educacion] [ESTABLECIMIENTO] [Inicio] [Fin] [Titulo] [ESPECIALIDAD]
Informacion de Idiomas:
IDIOMA CONVERSACION ESCRITURA LECTURA
[IDIOMA] [CONVERSACION] [ESCRITURA] [LECTURA]
Experiencias Laborales:
Telefono: [TELEFONO]
Jefe: [JEFE] Puesto: [PUESTO] Salario: [SALARIO]
Fecha Ingreso: [FECHA_INGRE Fecha Salida: [FECHA_RETIR Referencia: [REFE
Motivo retiro: [MOTIVO_RETIRO] Descripcion: [DESCRIPCION]
Informacion Socioeconomica:
[AGRUPACI [DETALLE_/ [DEPENDIEI [DETALLE_I
[VIVIENDA] [PAGO_VIVI [FLAG_DEU [MONTO_DE
[MONTO_DE [MONTO_DE [FLAG_OTR [MONTO_OT
[FUENTES_ [VEHICULO] [TIPO_VEHIC [MODELO_V
[PLACA_VE
Grupos de filas
Grupos de columnas
Salida
Consola del Administrador de paquetes Lista de errores Lista de tareas Salida
Listo
0 / 0


Archivo Editar Ver Git Proyecto Compilar Depurar Formato Prueba Analizar Herramientas Extensiones Ventana Ayuda Buscar
Debug Any CPU Iniciar
[Copilot] To enable GitHub Copilot, add your GitHub account to Visual Studio Add GitHub Account Don't show this again
Datos del informe
Nuevo Editar...
Campos integrados
Parámetros
Imágenes
Orígenes de datos
Conjuntos de datos
Perssona
Id
Foto
Empleado
lugarFechaNacimiento
NACIONALIDAD
Edad
EstadoCivil
Genero
DPI
NIT
IGSS
RELIGION
DEPARTAMENTO
MUNICIPIO
LICENCIA
Datos del informe Explorador de soluciones
Cuadro de herramientas
Búsqueda en el Cuadro de herramientas
Elementos de informe
Puntero
Cuadro de texto
ReportePersona.rdlc [Diseño] frmProfesion.cs [Diseño] Vacaciones.cs [Diseño] frmVacaciones...
[DISCAPACIDA [DROGAS] [ALCOHOL] [FUMA] [PESO]
[ESTATURA] [SANGRE] [PASATIEMPO: [DEPORTES]
No: EMPRESA TELEFONO RELACION
[EMPRESA] [TELEFONO] [RELACION]
No: NOMBRE TELEFONO RELACION
[NOMBRE] [TELEFONO] [RELACION]
[TRABAJOENC [SOLICITUDEN «Expr»
[PLAZA] [DISPONIBILID/ [FAMILIAR_EN [FAMILIAR_EN [ENCA_RELAC
[PUESTO_CON [PLAZA_VACAI
NO: EMERGENCIAS PARENTESCO TELEFONO
[EMERGENCIAS] [PARENTESCO] [TELEFONO]
Grupos de filas
Grupos de columnas
Salida
Consola del Administrador de paquetes Lista de errores Lista de tareas Salida
Listo
0 / 0

Ya teniendo el diseño se hace una ventana para visualizar este reporte para la vista del usuario.



Programación al darle clic sobre el botón.



Con este botón se cargará el diseño del reporte programación del botón.

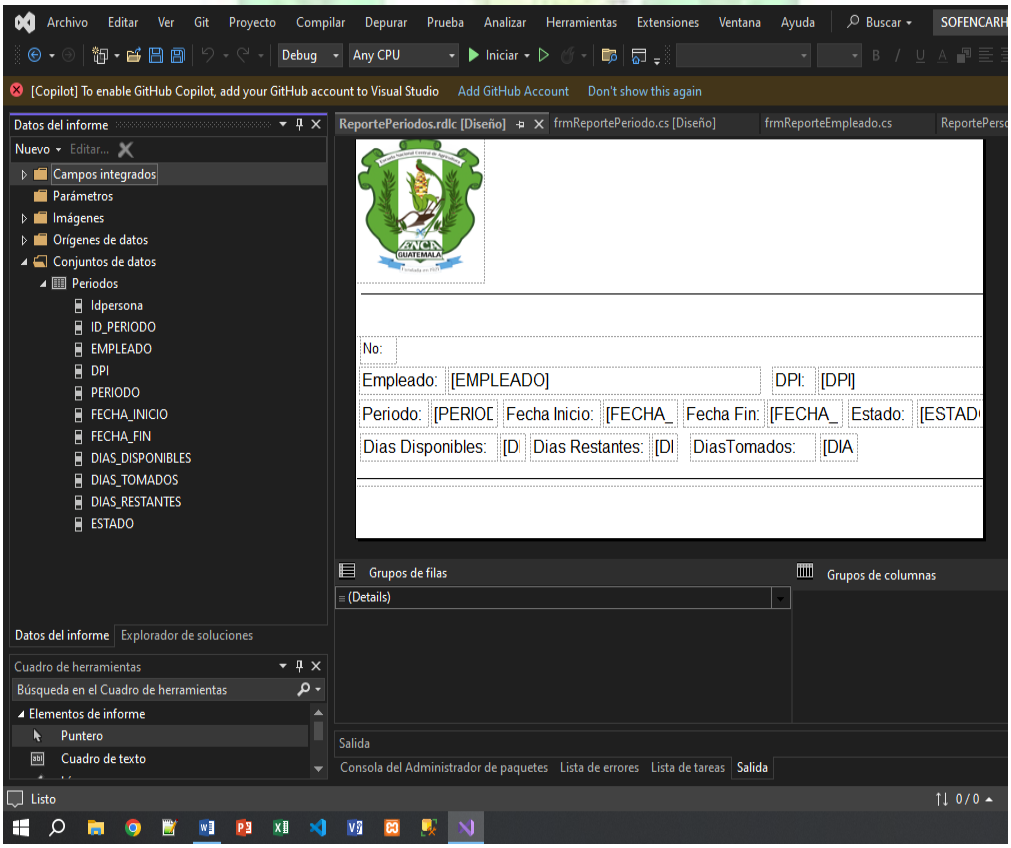
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
using System.Windows.Forms;

namespace Capa_Vista.Reporte
{
    public partial class frmReporteEmpleado : Form
    {
        public int Idpersona {  get; set; }
        public frmReporteEmpleado()
        {
            InitializeComponent();
        }

        private void frmReporteEmpleado_Load(object sender, EventArgs e)
        {



        }

        private void pictureBox3_Click(object sender, EventArgs e)
        {
            try
            {


this.sp_ListarPersonaReporteTableAdapter.Fill(this.personaDataSet.sp_ListarPersonaRe
porte, Idpersona);

this.sp_ListarHomeUbicacionTableAdapter.Fill(this.personaDataSet.sp_ListarHomeUbicac
ion, Idpersona);

this.sp_ListarHomeFamiliaTableAdapter.Fill(this.personaDataSet.sp_ListarHomeFamilia,
Idpersona);

this.sp_listarHomeEducacionTableAdapter.Fill(this.personaDataSet.sp_listarHomeEducac
ion,Idpersona);

this.sp_ListarHomeidiomaTableAdapter.Fill(this.personaDataSet.sp_ListarHomeidioma,
Idpersona);

this.sp_ListarHomeRHEXPERIENCIALABORALTableAdapter.Fill(this.personaDataSet.sp_Lista
rHomeRHEXPERIENCIALABORAL, Idpersona);

this.sp_ListarHomeRHSOCIOECONOMICOTableAdapter.Fill(this.personaDataSet.sp_ListarHom
eRHSOCIOECONOMICO, Idpersona);

this.sp_ListarHomeRHFISICABIOLOGICATableAdapter.Fill(this.personaDataSet.sp_ListarHo
meRHFISICABIOLOGICA, Idpersona);

this.sp_ListarHomeReferenciaLaboralTableAdapter.Fill(this.personaDataSet.sp_ListarHo
meReferenciaLaboral, Idpersona);

this.sp_ListarHomeReferenciaPersonalTableAdapter.Fill(this.personaDataSet.sp_ListarH
omeReferenciaPersonal, Idpersona);

this.sp_ListarHomeRHOTROSDATOSTableAdapter.Fill(this.personaDataSet.sp_ListarHomeRHO
TROSDATOS, Idpersona);
```

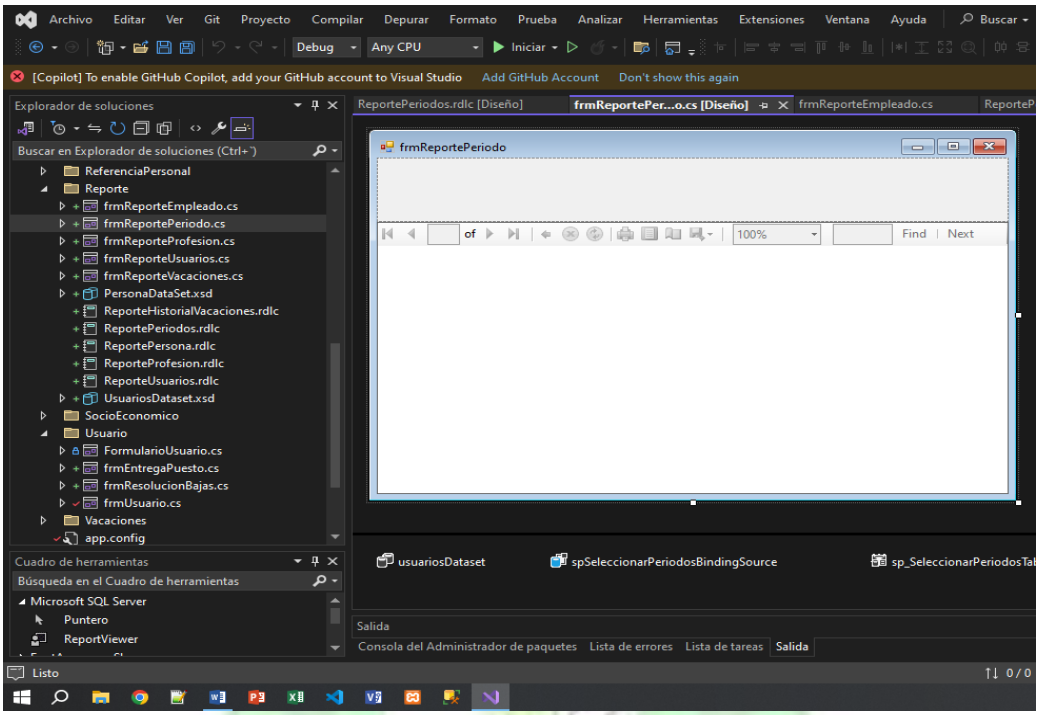
```
this.sp_ListarHomeRHDATOSADICIONALESTableAdapter.Fill(this.personaDataSet.sp_ListarH
omeRHDATOSADICIONALES, Idpersona);

                this.reportViewer1.RefreshReport();

            }
            catch(Exception ex)
            {
                this.reportViewer1.RefreshReport();
            }
        }
    }
    }
```

Aquí se hace el llamado a cada procedimiento para que se pueda visualizar en el reporte.

Reporte de periodos. es la misma lógica solo que cambia el procedimiento almacenado.



Aquí hacemos el diseño del reporte y luego creamos una vista para el usuario mediante la cual se va referenciar al reporte.

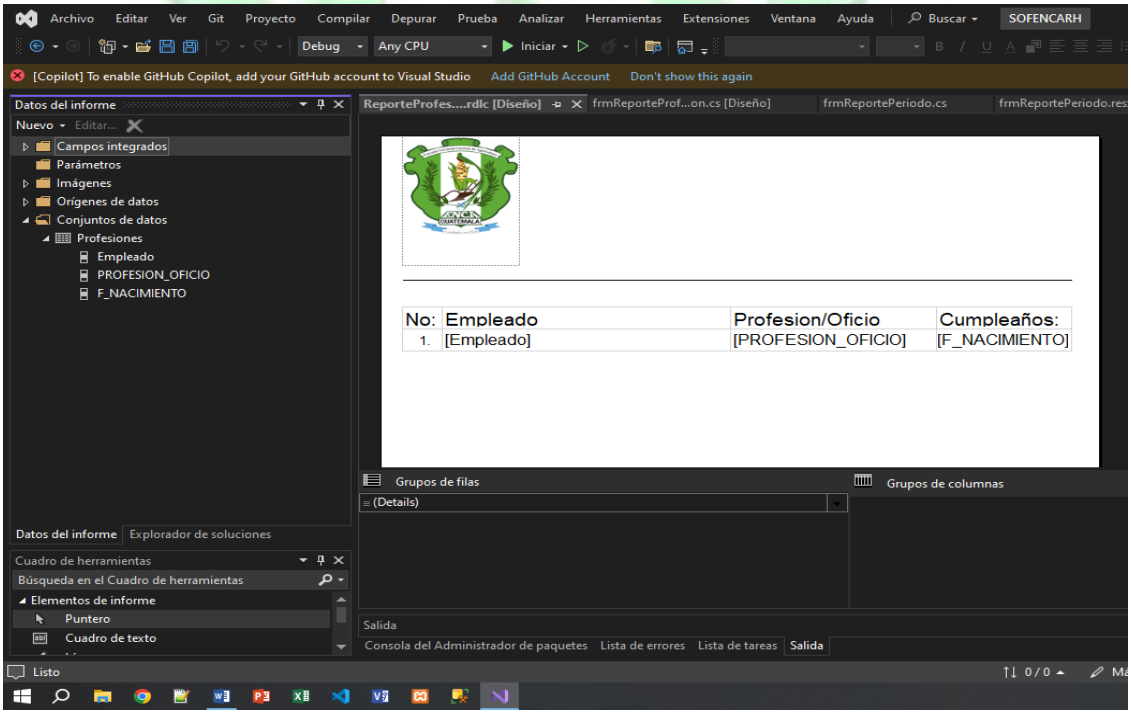


Programación.

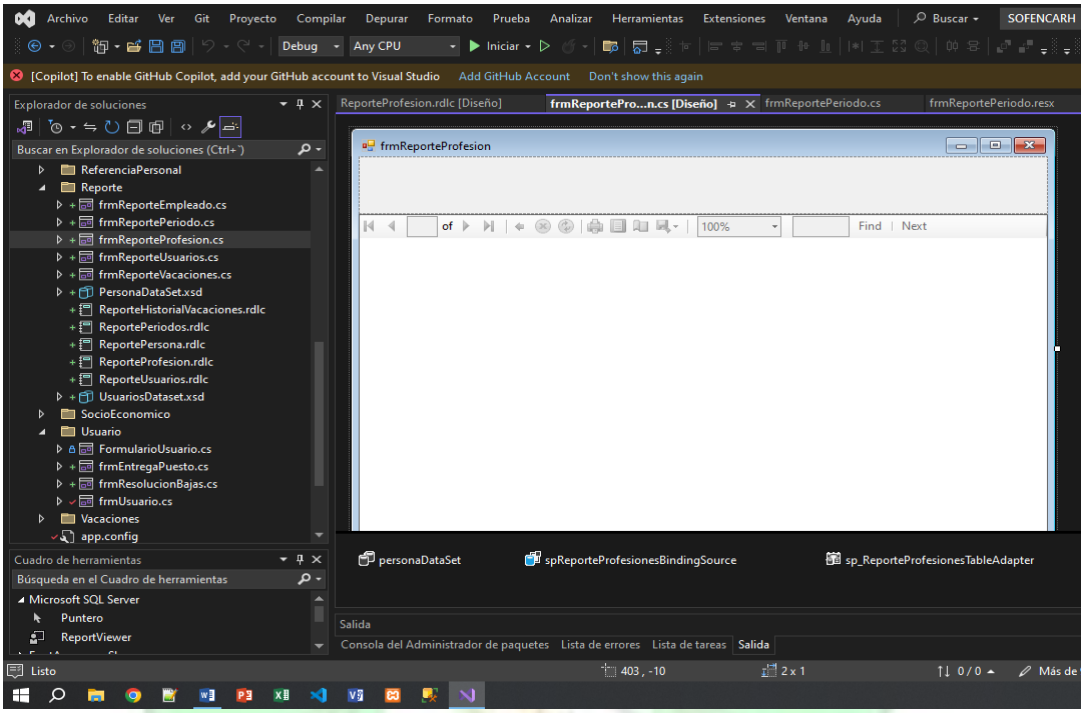
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Capa_Vista.Reporte
{
    public partial class frmReportePeriodo : Form
    {
        public int Idpersona {  get; set; }
        public frmReportePeriodo()
        {
            InitializeComponent();
        }

        private void frmReportePeriodo_Load(object sender, EventArgs e)
        {
            try
            {
```

```
this.sp_SeleccionarPeriodosTableAdapter.Fill(this.usuariosDataset.sp_SeleccionarPeri
odos, Idpersona);
                this.reportViewer1.RefreshReport();
            } catch (Exception ex) {
                this.reportViewer1.RefreshReport();
            }


        }
    }
    }
```

Reporte de Profesiones empleados.



Se crea el diseño y toda lógica para el reporte, y luego se hace una vista para este reporte donde se va cargar el reporte.

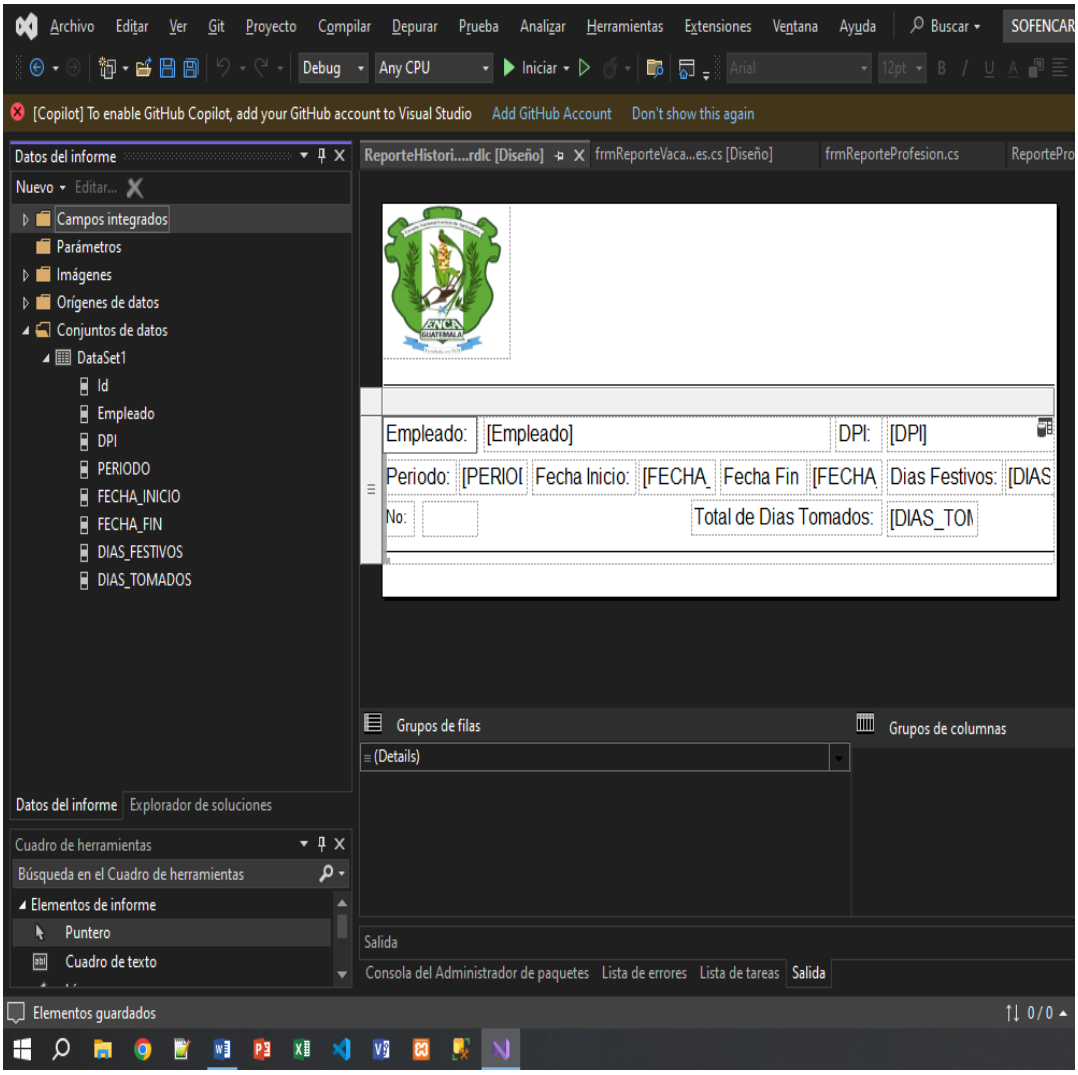


Programación.

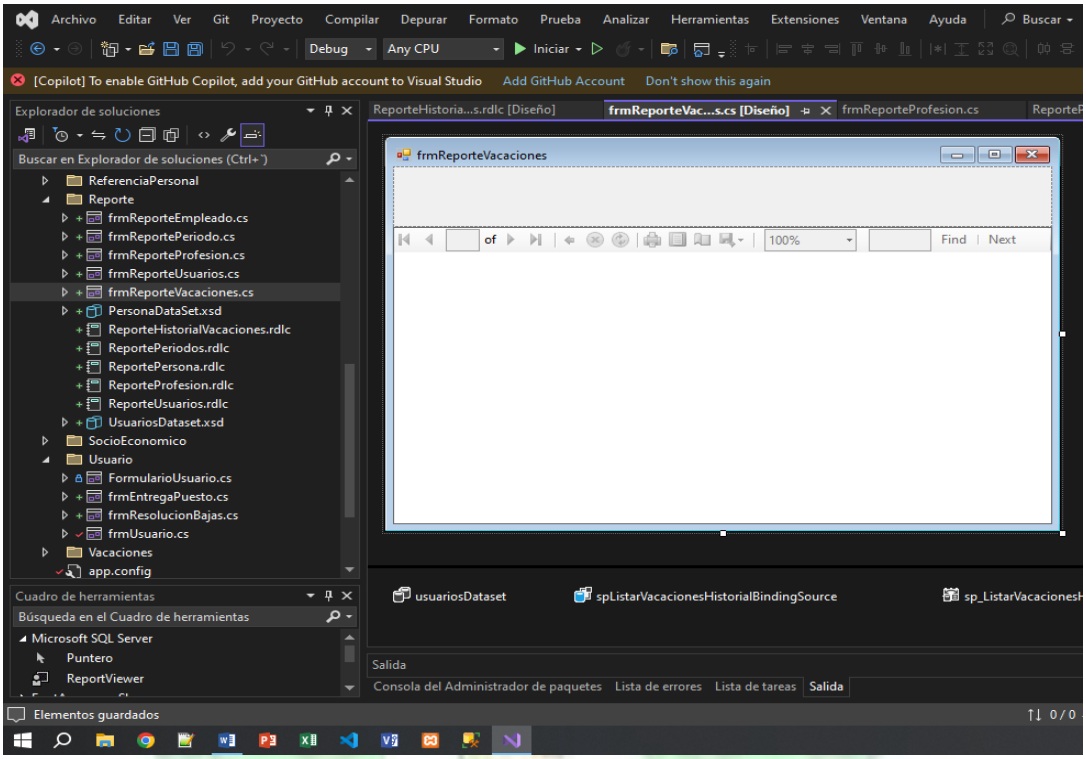
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Capa_Vista.Reporte
{
    public partial class frmReporteProfesion : Form
    {
        public frmReporteProfesion()
        {
            InitializeComponent();
        }

        private void frmReporteProfesion_Load(object sender, EventArgs e)
        {
            try
            {

this.sp_ReporteProfesionesTableAdapter.Fill(this.personaDataSet.sp_ReporteProfesione
s);
                this.reportViewer1.RefreshReport();
            }
            catch (Exception ex )
```

```
        {
            this.reportViewer1.RefreshReport();
        }
    }
}
}
```

Reporte de asignar vacaciones.



Igual se crea una vista para este informe para cargar los datos.

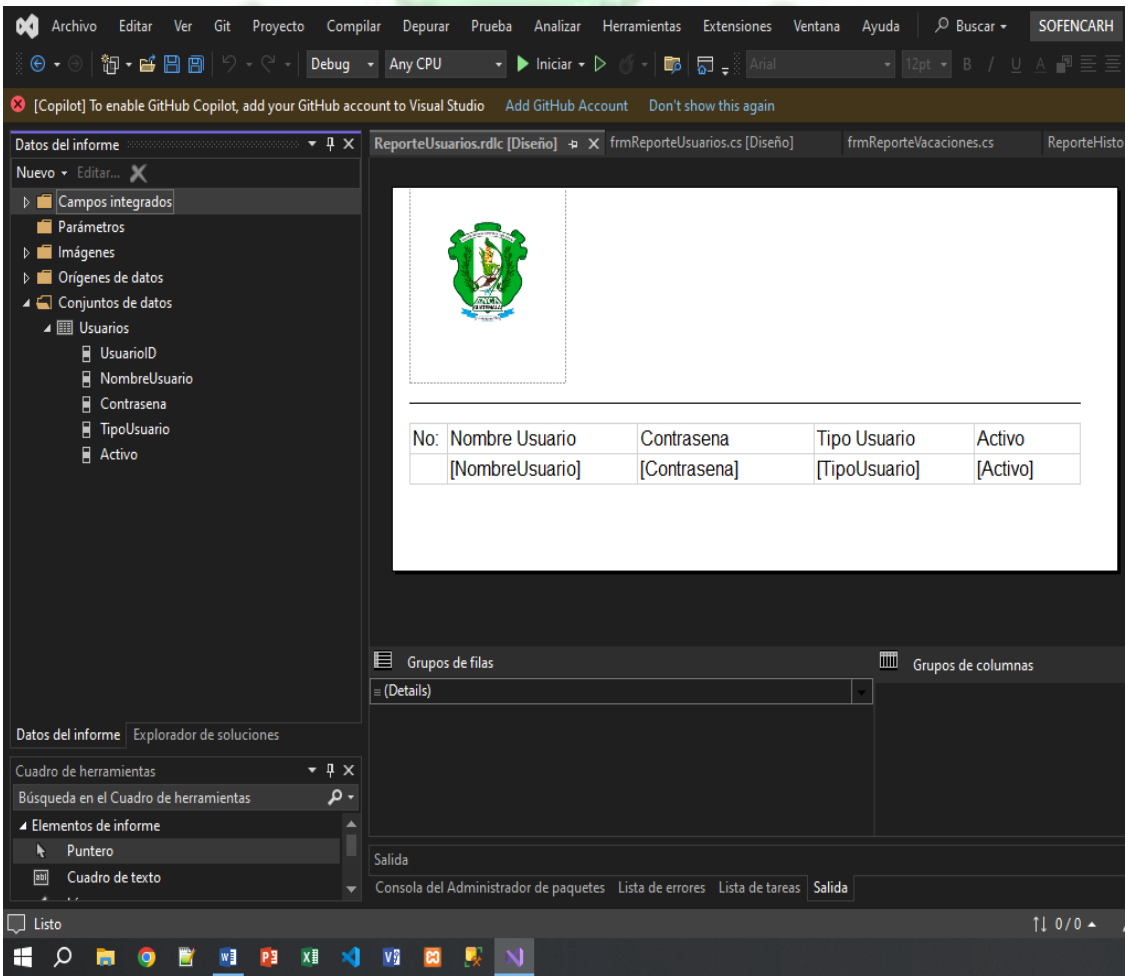


Programación.

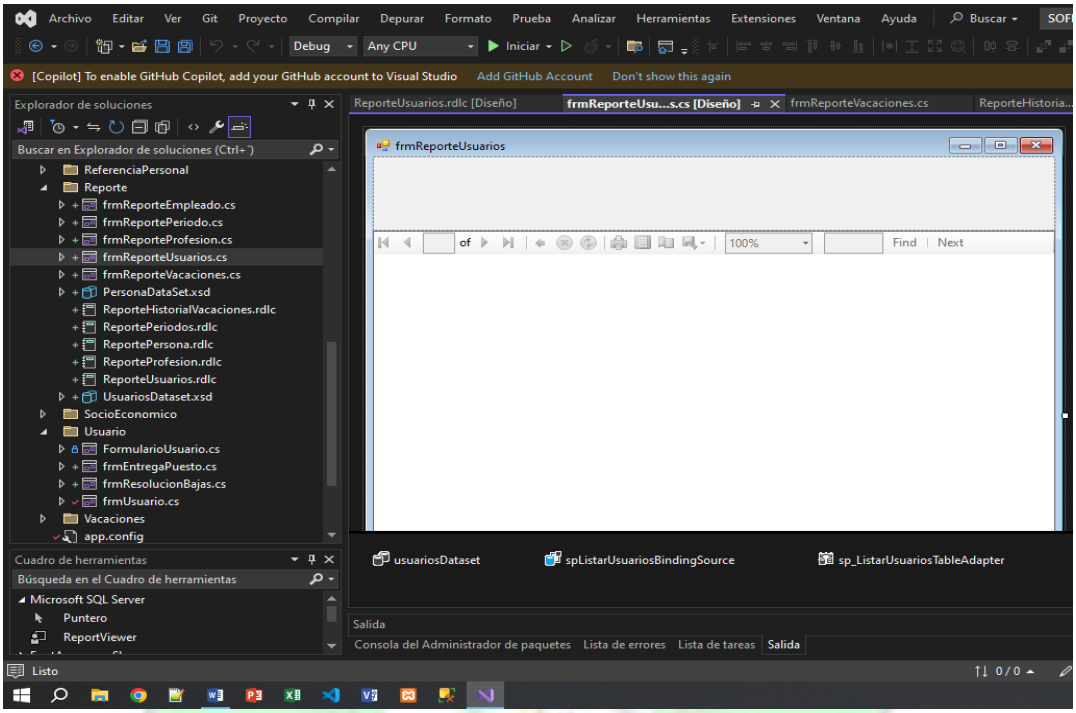
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Capa_Vista.Reporte
{
    public partial class frmReporteVacaciones : Form
    {
        public int Idpersona {  get; set; }
        public frmReporteVacaciones()
        {
            InitializeComponent();
        }

        private void frmReporteVacaciones_Load(object sender, EventArgs e)
        {
            try {

this.sp_ListarVacacionesHistorialTableAdapter.Fill(this.usuariosDataset.sp_ListarVac
acionesHistorial, Idpersona);
                this.reportViewer1.RefreshReport();
            }
```

```
        catch (Exception ex)
        {
            this.reportViewer1.RefreshReport();
        }

    }

    private void panel1_Paint(object sender, PaintEventArgs e)
    {

    }
}
}
```

Reporte de usuarios.



Al tener el diseño del reporte de crea una vista para mostrar este reporte.



Programación.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Capa_Vista.Reporte
{
    public partial class frmReporteUsuarios : Form
    {
        public frmReporteUsuarios()
        {
            InitializeComponent();
        }

        private void frmReporteUsuarios_Load(object sender, EventArgs e)
        {
            try
            {   // TODO: This line of code loads data into the
'usuariosDataset.sp_ListarUsuarios' table. You can move, or remove it, as needed.

this.sp_ListarUsuariosTableAdapter.Fill(this.usuariosDataset.sp_ListarUsuarios);

                this.reportViewer1.RefreshReport();
            }
```

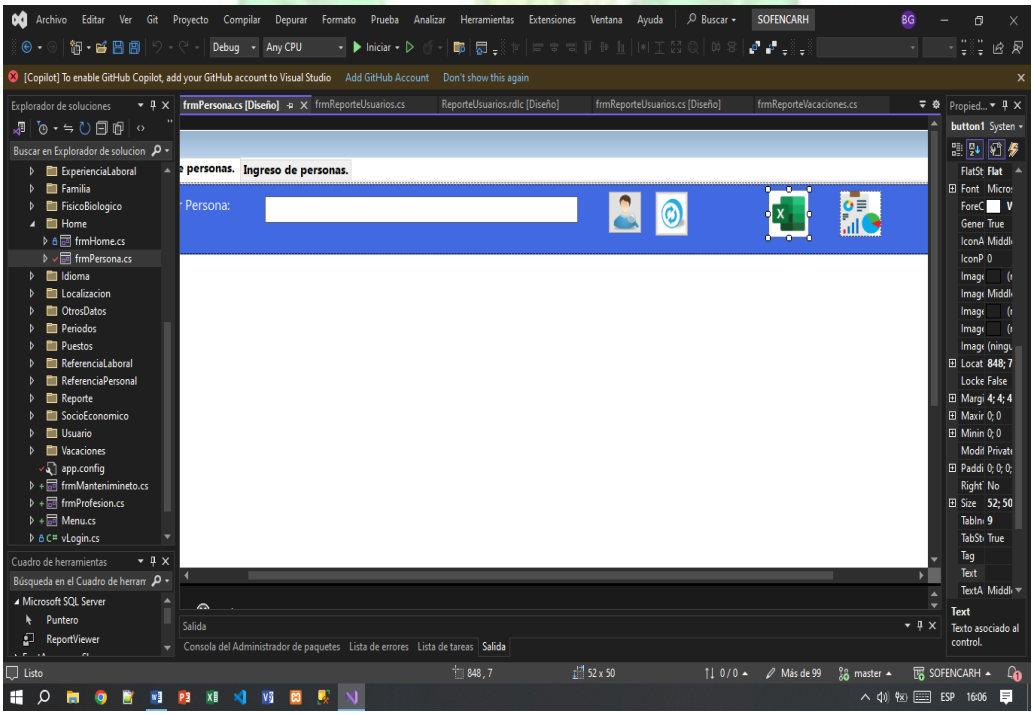
```
            catch (Exception ex)
            {
                this.reportViewer1.RefreshReport();
            }


        }
    }
    }
```

Creación de archivos Excel de datos del Sistema.

En esta parte se crea una programación la cual nos permite generar archivos Exceles de los datos que se tienen ingresados en los datagrid del sistema.

Para ello se hace la siguiente lógica, en la parte de cada ventana de vista que se va realizar el archivo Excel se debe de diseñar un botón el cual creara este archivo.

Como podemos ver aquí en persona.



Esta el método el cual se debe de programar y la programación es la siguiente,

Programación para crear archivo Excel.

```
private void button1_Click(object sender, EventArgs e)
{
    ExportarDataGridViewAExcel(dtgPersona);
}

private void ExportarDataGridViewAExcel(DataGridView dataGridView)
{
    // Verificar si hay datos en el DataGridView
    if (dataGridView.Rows.Count == 0)
    {
        MessageBox.Show("No hay datos para exportar.", "Exportar a Excel",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    // Crear un nuevo libro de Excel
    var workbook = new XLWorkbook();
    var worksheet = workbook.Worksheets.Add("Hoja1");

    // Agregar los encabezados de columnas
    for (int i = 1; i <= dataGridView.Columns.Count; i++)
    {
        worksheet.Cell(1, i).Value = dataGridView.Columns[i - 1].HeaderText;
    }

    // Agregar los datos de las filas
    for (int i = 0; i < dataGridView.Rows.Count; i++)
    {
        for (int j = 0; j < dataGridView.Columns.Count; j++)
        {
            worksheet.Cell(i + 2, j + 1).Value =
dataGridView.Rows[i].Cells[j].Value?.ToString();
        }
    }

    // Guardar el archivo de Excel
    SaveFileDialog saveFileDialog = new SaveFileDialog();
    saveFileDialog.Filter = "Archivos Excel (*.xlsx)|*.xlsx";
    saveFileDialog.FileName = "ArchivoExcel.xlsx";
    if (saveFileDialog.ShowDialog() == DialogResult.OK)
    {
        try
        {
            workbook.SaveAs(saveFileDialog.FileName);
            MessageBox.Show("El archivo Excel se ha exportado correctamente.",
"Exportar a Excel", MessageBoxButtons.OK, MessageBoxIcon.Information);

            // Abrir el archivo de Excel después de guardarlo
            System.Diagnostics.Process.Start(saveFileDialog.FileName);
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error al guardar el archivo Excel: " + ex.Message,
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
```

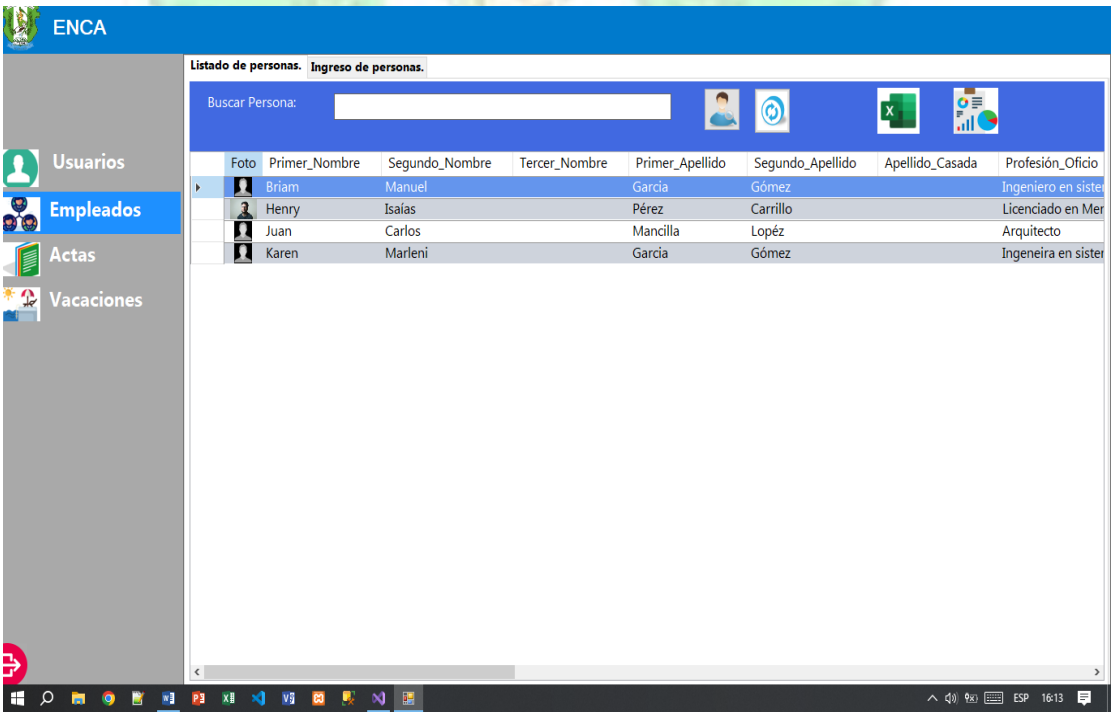
```
}
    }
```

Esta es la programación mediante dos funcione una que lleva toda la lógica de programación y la otra la cual hace referencia la función mediante el botón.

Y esto es en todas las ventanas solo que cambia el nombre del en la programación del datagrid del cual se está seleccionando los datos.

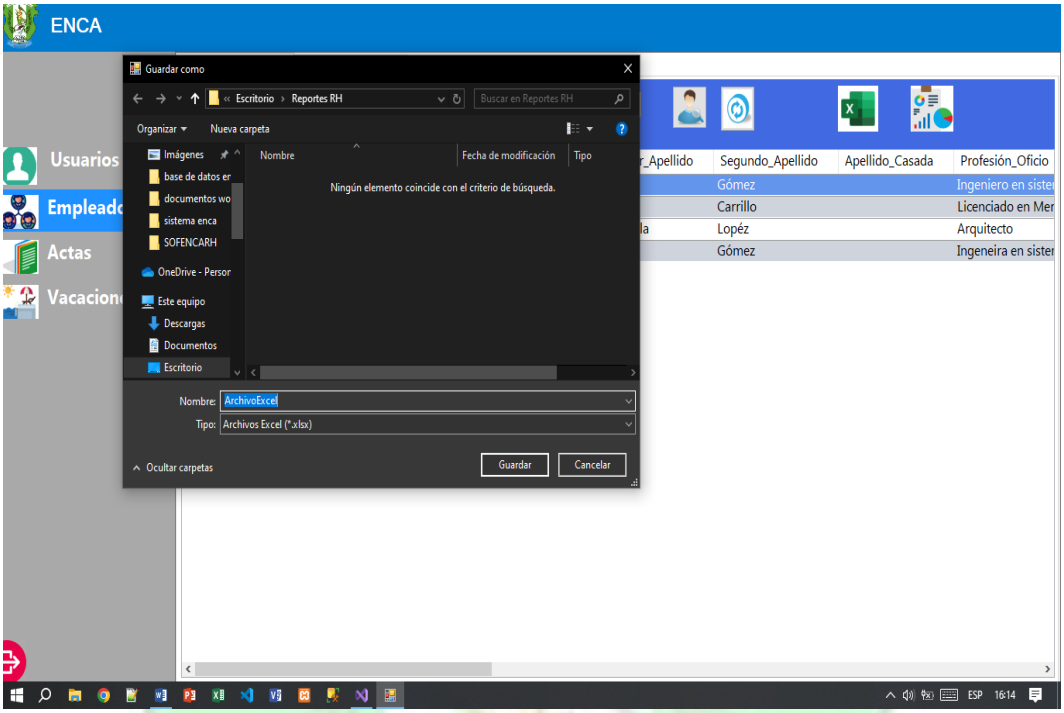
Ejecución del programa para ver la funcionalidad del método de creación de archivos Excel con la información de los datos del empleado, las actas los usuarios, etc.

Para ello vamos al menú de empleados.



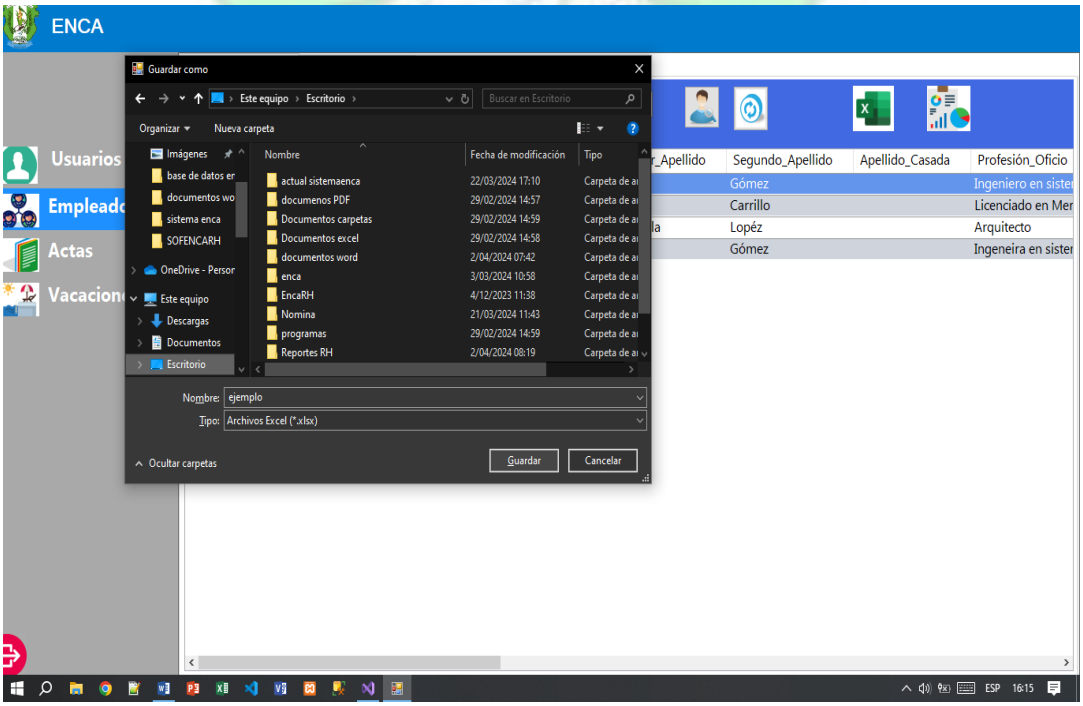
Vamos al botón de Excel.

Al dar clic sobre el botón nos abre el gestor de archivos de nuestro sistema operativo.

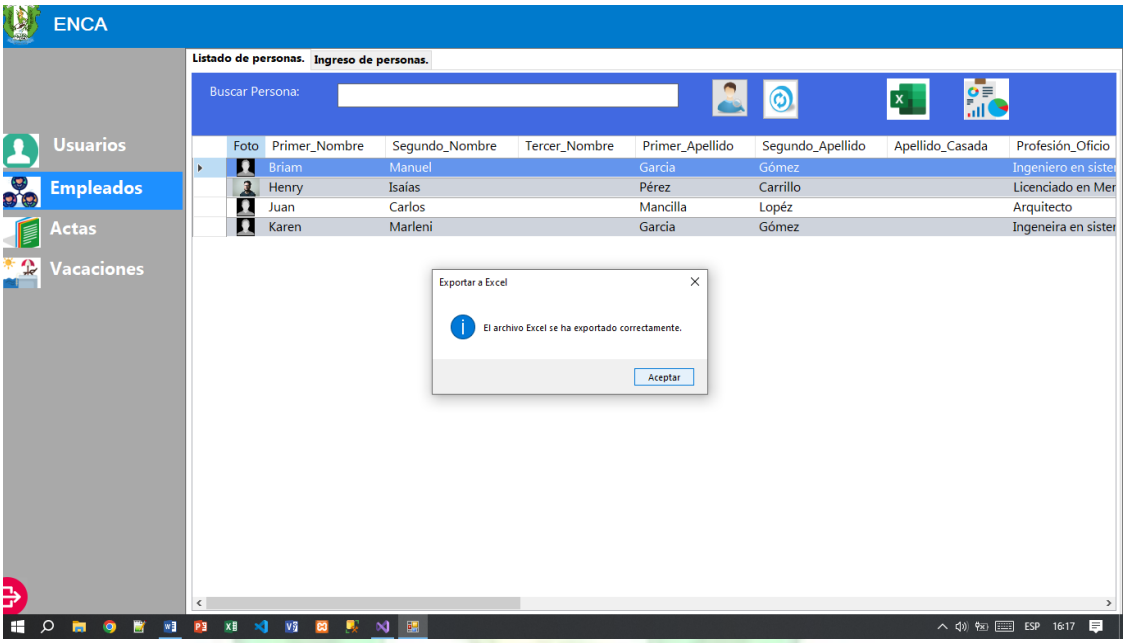


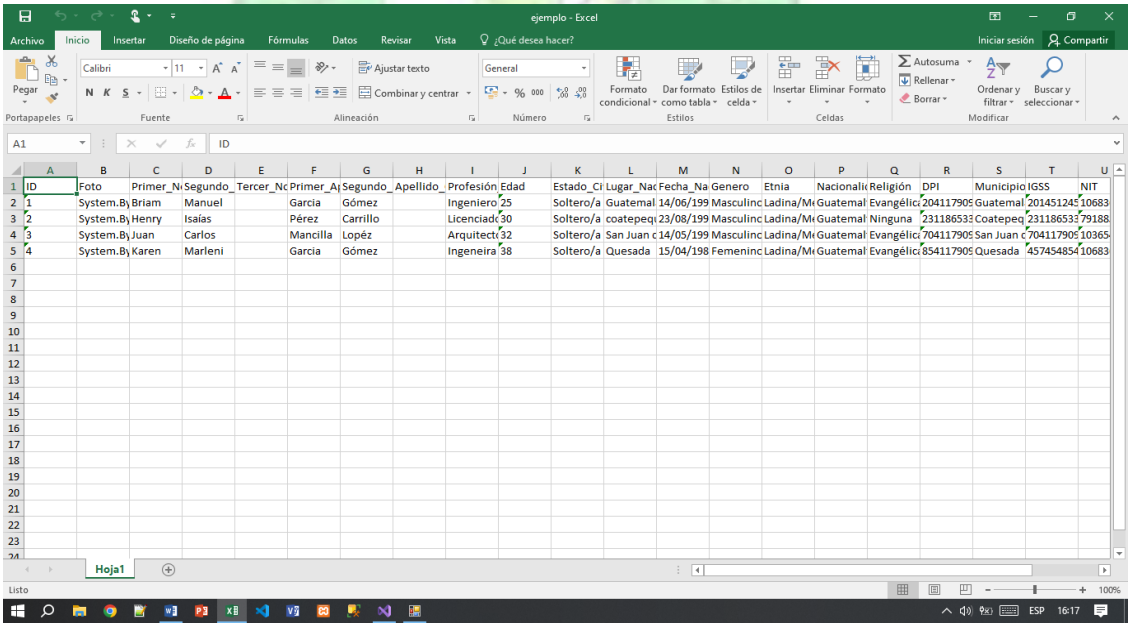
Aquí tenemos que ver donde se guardara nuestro archivo Excel con los datos, tenemos que darle un nombre y una ubicación.

En esta demostración le puse ejemplo al archivo y se va guardar en escritorio.



Lo guardamos al guardarlo en base a la programación que se le dio esperamos un cierto momento y el mismo archivo se abre de forma automática.

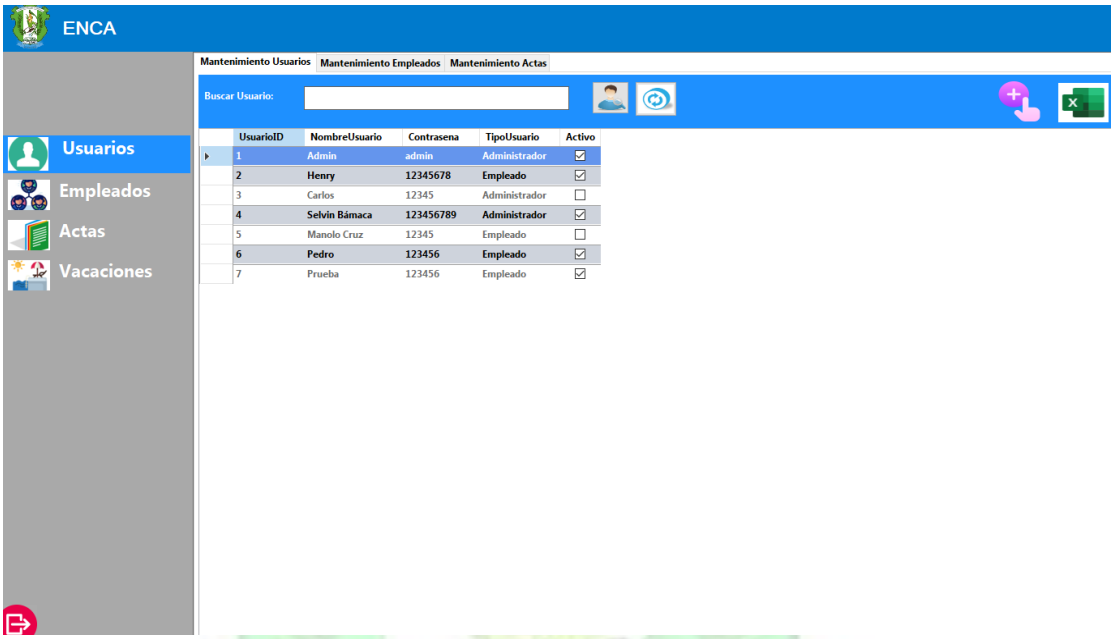


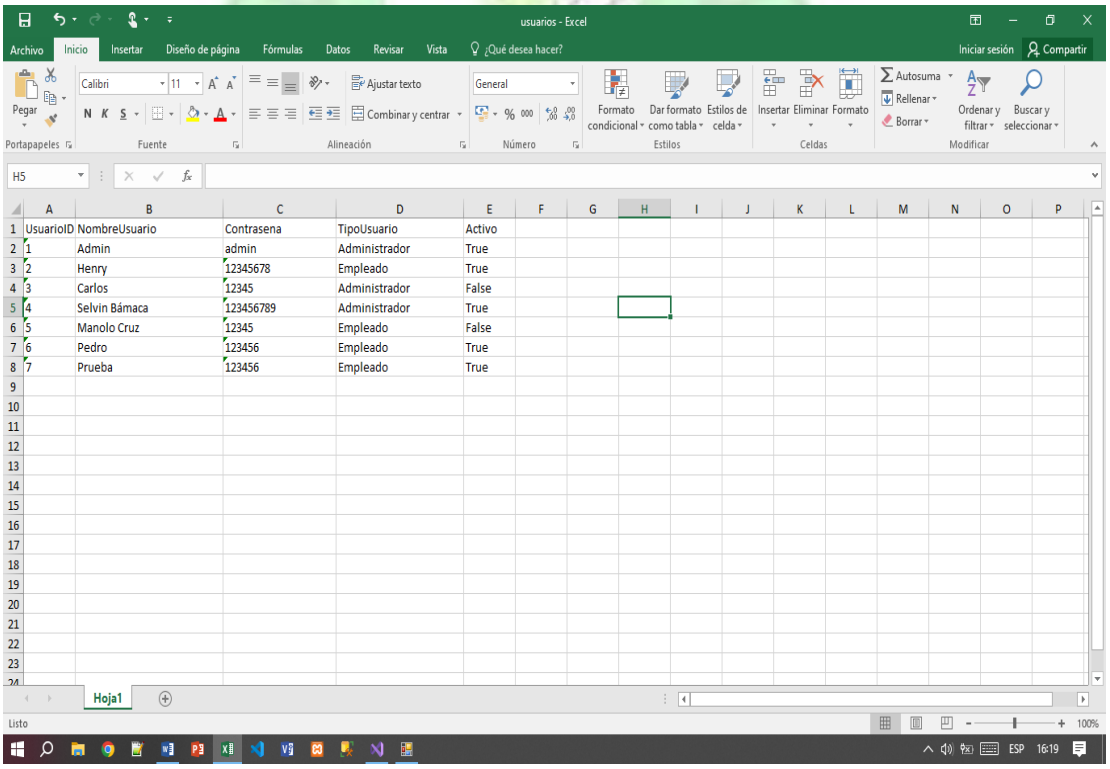


Y como podemos ver nos abre el archivo Excel de forma automática.

Y esta es la lógica en cada uno de las interfaces que tiene el botón de Excel.

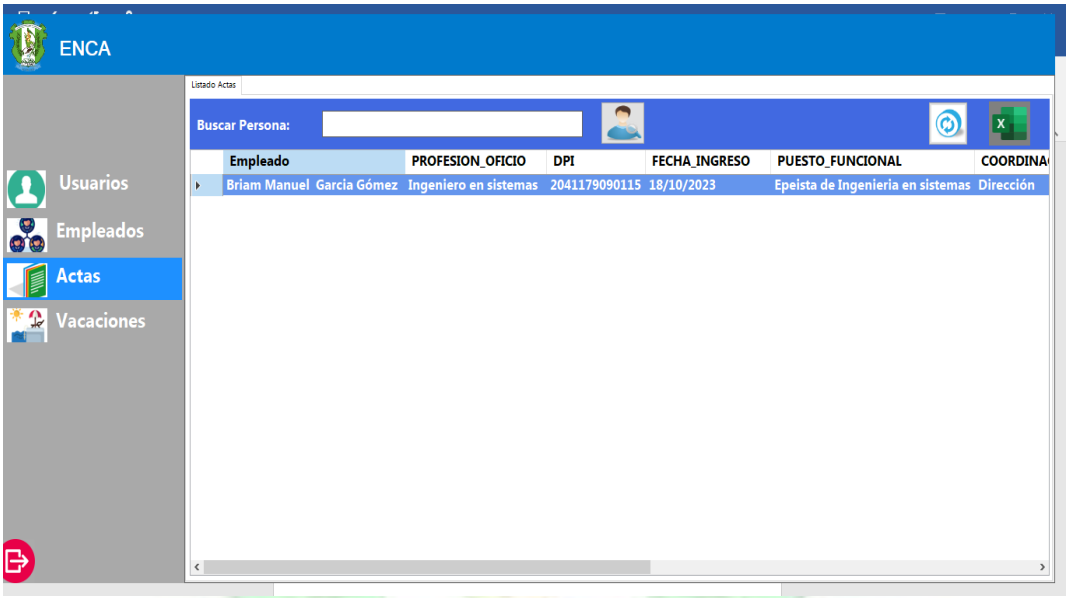
Vamos a usuarios.



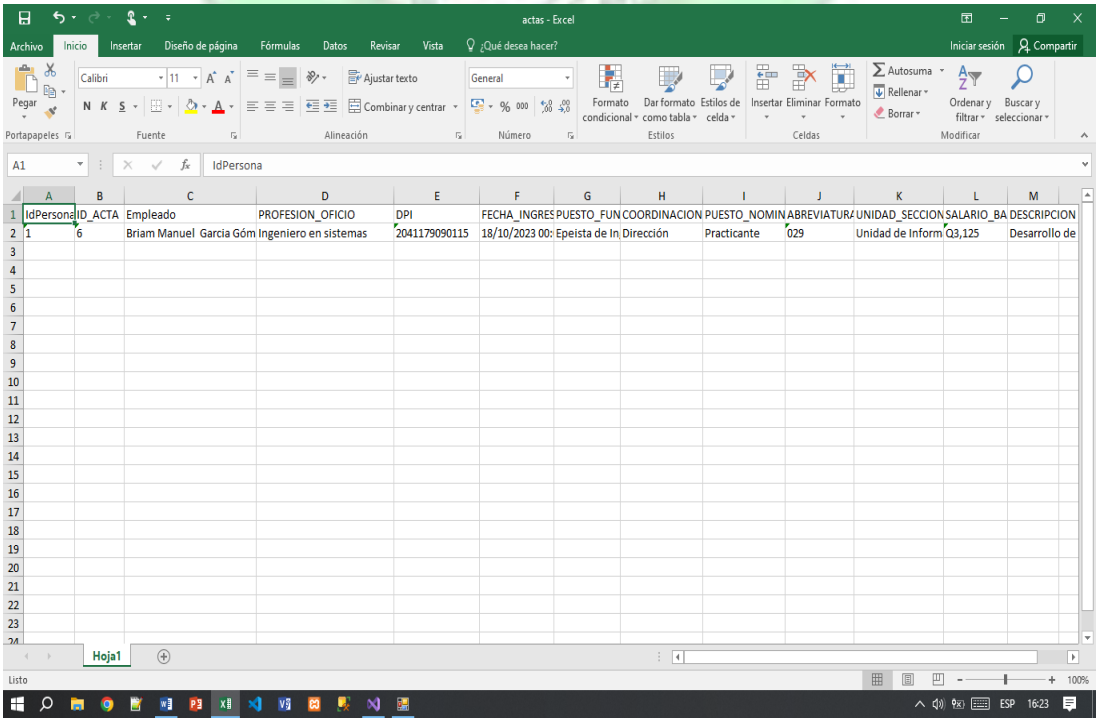


Y nos creó el archivo de forma automática.

En actas de igual manera.



Seleccionamos el botón de Excel damos clic guardamos el archivo y nos muestra nuestro archivo Excel con los datos.



Módulo de Vacaciones.

En este módulo se manejarán la Asignación de periodos de vacaciones, asignación de solicitud de vacaciones y la cancelación de vacaciones esto por empleado, como primera característica importante se debe de tener en cuenta algunos parámetros que implican el desarrollo y programación de este módulo en base a las necesidades que se requieren.

Primero: Que en los primeros 5 años que un empleado lleve trabajando en la ENCA la cantidad de días de vacaciones son de 23 y después de los 5 años cumplidos de estar trabajando en la ENCA, el empleado tiene derecho a 25 días de vacaciones por periodo cumplido que es igual a un año calendario.

Segundo: Que se tiene que llevar la suma total de los días acumulados que lleva un empleado en base a sus periodos, esto para darnos una cantidad exacta de los días que lleva el empleado acumulado en base a sus periodos.

Tercero: Que al asignar vacaciones en base a sus periodos del empleado se debe de tener en cuenta las siguientes características, que los días sábados y domingos no se toman y los días feriados, asuetos o festividades de la ENCA, sino solo aquellos días que si son oficiales laborales para la ENCA y que también no se puede asignar más días de los que tiene la persona en base a cada periodo, si esto sucede se tomaran días del siguiente periodo.

Cuarto: Que cuando un empelado es nuevo o se asignara un periodo de vacaciones en base a un año anterior al actual se debe de respetar la siguiente condición que el empleado tiene derecho a vacaciones después de 150 días laborados en la ENCA,

hasta que cumpla esa cantidad de días laborales tiene el derecho a vacaciones, esto cuando un empleado es nuevo o se está asignando un nuevo periodo a un empleado que ya tiene más de un año de estar trabajando en la ENCA.

Quinto: Que también se tiene la funcionalidad de poder cancelar las vacaciones si así se solicitara al empleado, esto conlleva a poder tomar los días que lleva a la fecha de vacaciones tomarlos y restarlos a los días acumulados en base al periodo que se le asignaron y actualizar la resta y suma de esos datos.

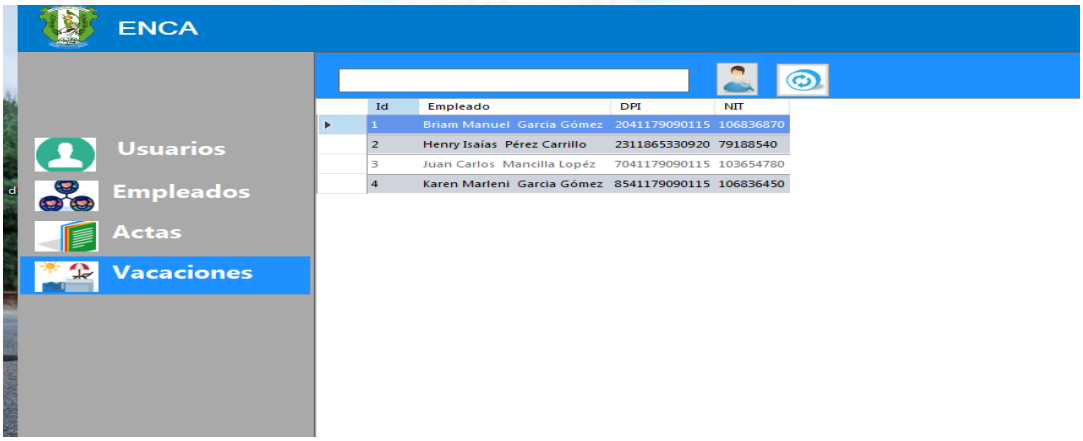
Sexto: Que se necesitan los reportes de asignación de periodos de vacaciones, solicitud de vacaciones y cancelación de vacaciones entre otros.

Teniendo esto en cuenta emperraremos con la programación de este módulo.

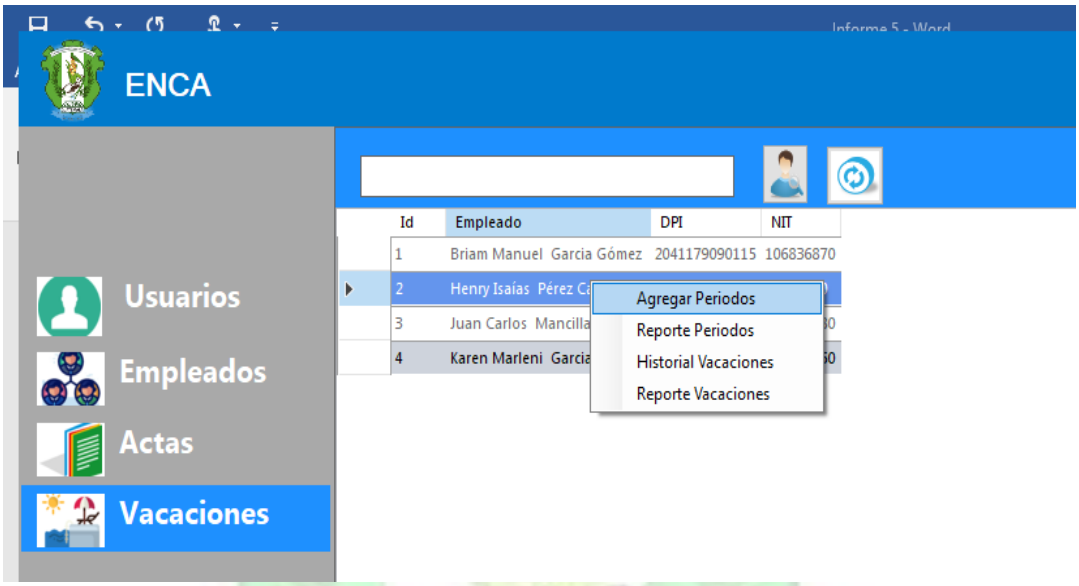
Asignación de periodos de vacaciones.

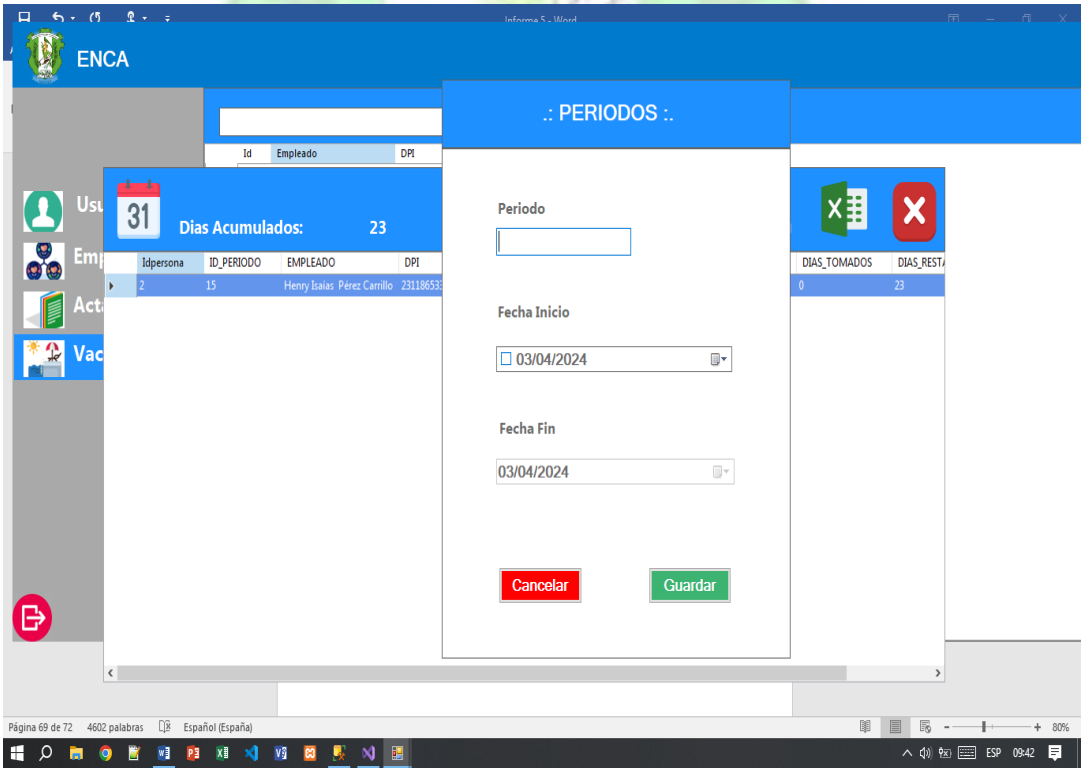
Para esta parte del módulo tenemos que ver el registro de la persona en base al ingreso de un empleado al sistema, esto para poder asignar los periodos por empleado.

En base a esta lógica se creó el menú de vacaciones en el sistema el cual despliega los empleados solo con su nombre completo su número de DPI y NIT.

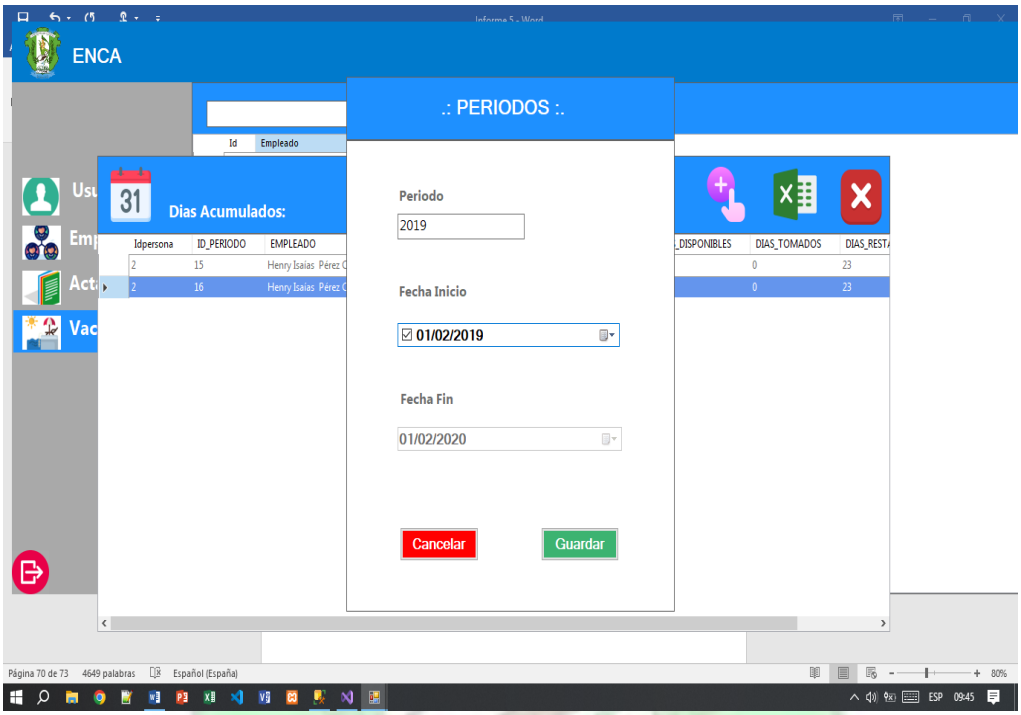


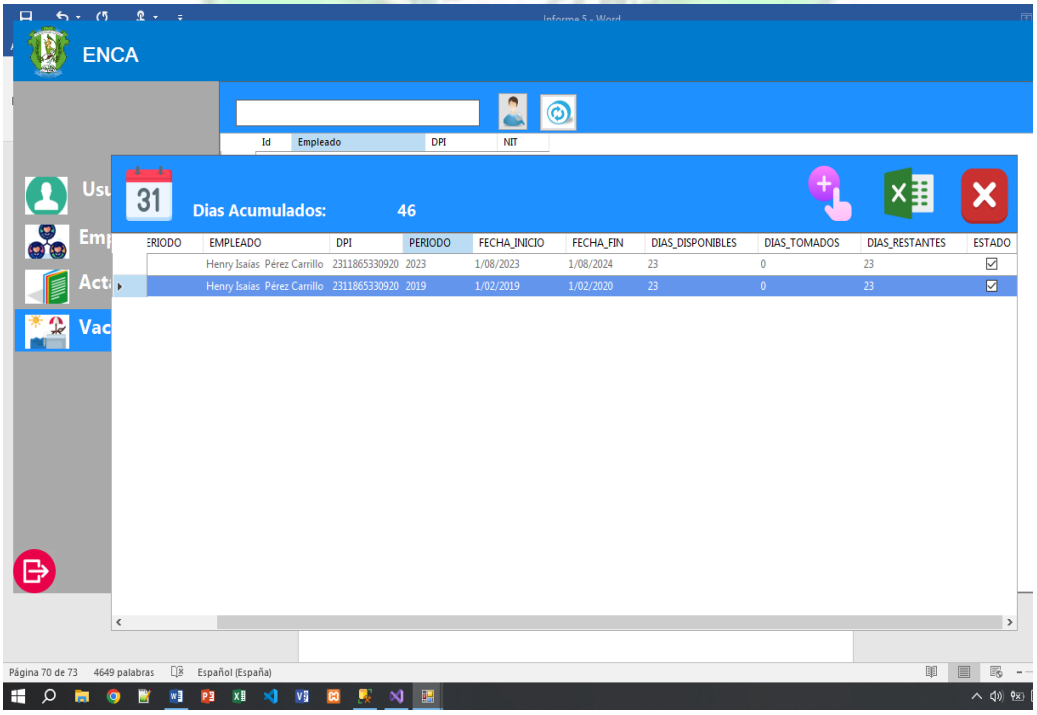
La funcionalidad es la siguiente se selecciona el registro del empleado al cual se le va asignar sus periodos de vacaciones, damos clic derecho sobre este empleado y nos aparece un menú seleccionamos agregar periodos.



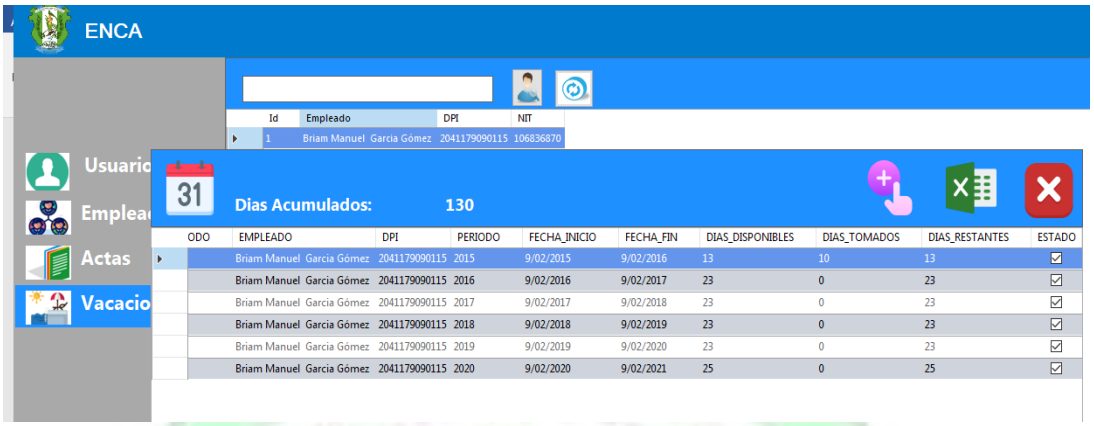


Aquí lo único que se debe de ingresar es el periodo al cual se le va asignar el empleado y la fecha de inicio y el sistema hace automático lo siguiente, pone la fecha fin, los días acumulados, días tomados y días restantes en base al periodo.





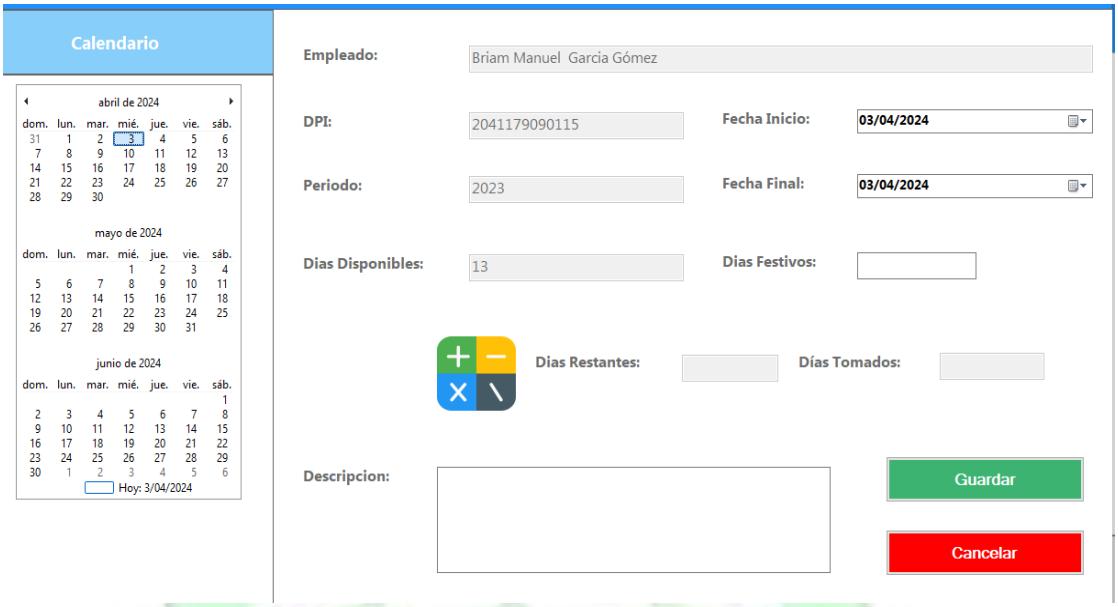
En base a esta lógica se ingresan más de 5 pedidos para ver la condición que después de 5 periodos son 25 días en vez de 23.



Como podemos ver este empleado tiene más de 5 periodos en este ejemplo y en el sexto periodo tiene 25.

En al agregar un periodo nuevo se hará un ejemplo.

En este caso se hace lo siguiente se asignó un periodo del 2023 con fecha 4 de septiembre y el sistema tiene que la fecha fin es el 4 de septiembre del 2024, en base a esto la condición toma los días laborados a la fecha por 365 dividido 365 y nos da los días de vacaciones que lleva al momento.



Como podemos ver solo lleva 13 días disponibles a la fecha de este periodo.

Y así.

Programación de asignación de periodos.

```
private void Mostrar()
{
    try
    {
        this.dtgAsignarPeriodos.DataSource = nPeriodos.MostrarPeriodos(idPersona);
        int suma = 0;

        foreach (DataGridViewRow fila in dtgAsignarPeriodos.Rows)
        {
            // Obtener el valor de la celda como una cadena de texto
            string dias = fila.Cells["DIAS_DISPONIBLES"].Value.ToString();

            // Intentar convertir la cadena a un entero antes de sumarlo
            int diasDisponibles;
            if (int.TryParse(dias, out diasDisponibles))
            {
                // Sumar el valor convertido a la suma total
                suma += diasDisponibles;
                lbTotalDias.Text = suma.ToString();
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Se produjo un error al intenta mostrar : {ex.Message}");
```

```
          }


private void dtgAsignarPeriodos_MouseClick(object sender, MouseEventArgs e)
 {
     if (e.Button == MouseButtons.Right)
     {
         var hti = dtgAsignarPeriodos.HitTest(e.X, e.Y);
         if (hti.RowIndex >= 0)
         {
             dtgAsignarPeriodos.Rows[hti.RowIndex].Selected = true;
         }
     }

     if (e.Button == MouseButtons.Right)
     {
         ContextMenuStrip menu = new System.Windows.Forms.ContextMenuStrip();
         int posicion = dtgAsignarPeriodos.HitTest(e.X, e.Y).RowIndex;
         if (posicion > -1)
         {

             menu.Items.Add("Asignar Vacaciones").Name = "Asignar Vacaciones" +
posicion;
             menu.Items.Add("Editar").Name = "Editar" + posicion;
         }
         menu.Show(dtgAsignarPeriodos, e.X, e.Y);
         menu.ItemClicked += new ToolStripItemClickedEventHandler(menuclick);
     }
 }
 private void menuclick(object sender, ToolStripItemClickedEventArgs e)
 {
     string id = e.ClickedItem.Name.ToString();
     try
     {

         if (id.Contains("Asignar Vacaciones"))
         {
             DateTime fechaActual = DateTime.Now;
             DateTime fechaPeriodo =
Convert.ToDateTime(this.dtgAsignarPeriodos.CurrentRow.Cells["FECHA_INICIO"].Value);
             // Calcular la diferencia en días entre las fechas
             TimeSpan diferencia = fechaActual - fechaPeriodo;
             // Verificar si la diferencia es de un año o más
             if (diferencia.Days >= 365)
             {
                 int diferenciaEnDias = diferencia.Days;
                 if (vacaciones == null || vacaciones.IsDisposed)
                 {

                     AbrirFormularioVacaciones(diferenciaEnDias,"Completo");
                 }
                 else
                 {
                     vacaciones.Activate();
                     // Si ya hay una instancia abierta, mostrar un mensaje de
advertencia
                     MessageBox.Show("Actualmente está ingresando un dato. No puede
actualizar un registro.");
                 }
```

```
            }
            else if(diferencia.Days >= 150)
            {
                int diferenciaEnDias = diferencia.Days;
                if (vacaciones == null || vacaciones.IsDisposed)
                {
                    AbrirFormularioVacaciones(diferenciaEnDias, "Incompleto");
                }
                else
                {
                    vacaciones.Activate();
                    // Si ya hay una instancia abierta, mostrar un mensaje de
advertencia
                    MessageBox.Show("Actualmente está ingresando un dato. No puede
actualizar un registro.");
                }
            }
            else
            {
                MessageBox.Show("No puedes asignar vacaciones a un empleado aun");
            }

        }

    }
    catch (Exception ex)
    {
        MessageBox.Show("Error al abrir el formulario: " + ex.Message);
    }
 }

 private void AbrirFormularioVacaciones(int diferencia, string Evento)
 {

    int IDPERIODO =
Convert.ToInt32(this.dtgAsignarPeriodos.CurrentRow.Cells["ID_PERIODO"].Value);
    string Empleado =
this.dtgAsignarPeriodos.CurrentRow.Cells["EMPLEADO"].Value.ToString();
    string DPI = this.dtgAsignarPeriodos.CurrentRow.Cells["DPI"].Value.ToString();
    string Periodo =
this.dtgAsignarPeriodos.CurrentRow.Cells["PERIODO"].Value.ToString();
    string Dias =
this.dtgAsignarPeriodos.CurrentRow.Cells["DIAS_DISPONIBLES"].Value.ToString();
    int DiasTomados =
Convert.ToInt32(this.dtgAsignarPeriodos.CurrentRow.Cells["DIAS_TOMADOS"].Value);


    vacaciones = new frmSolicitudVacaciones();
    vacaciones.FormClosed += (s, args) => { vacaciones = null; };
    vacaciones.IdPeriodo = IDPERIODO;
    vacaciones.IdPersona = idPersona;
    vacaciones.Evento = "Nuevo";
    vacaciones.diasTomados = DiasTomados;
    vacaciones.Acumulados = Convert.ToInt32(this.lbTotalDias.Text);
    vacaciones.CargarDatos(diferencia, Evento,Empleado, DPI, Periodo, Dias);
    vacaciones.ShowDialog();
 }
```

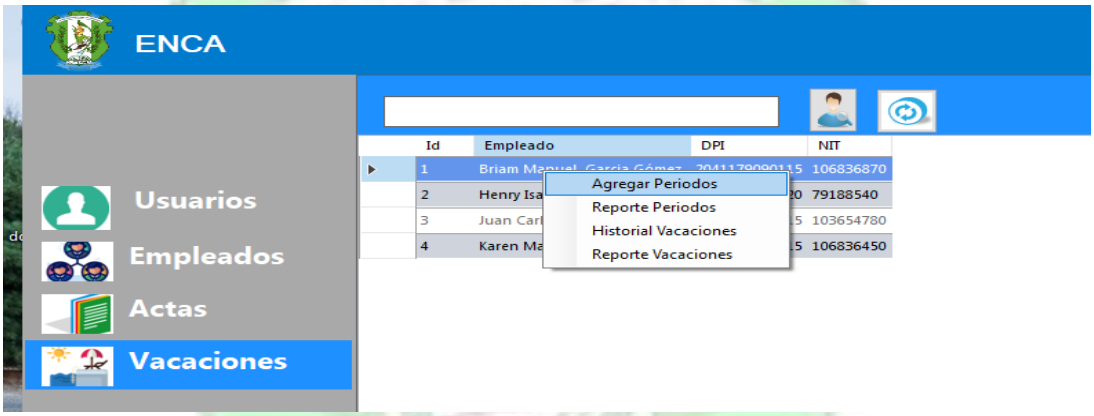
```
 private void pictureBox3_Click(object sender, EventArgs e)
 {
     ExportarDataGridViewAExcel(dtgAsignarPeriodos);
 }

 private void ExportarDataGridViewAExcel(DataGridView dataGridView)
 {
     // Verificar si hay datos en el DataGridView
     if (dataGridView.Rows.Count == 0)
     {
         MessageBox.Show("No hay datos para exportar.", "Exportar a Excel",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
         return;
     }
     // Crear un nuevo libro de Excel
     var workbook = new XLWorkbook();
     var worksheet = workbook.Worksheets.Add("Hoja1");
     // Agregar los encabezados de columnas
     for (int i = 1; i <= dataGridView.Columns.Count; i++)
     {
         worksheet.Cell(1, i).Value = dataGridView.Columns[i - 1].HeaderText;
     }
     // Agregar los datos de las filas
     for (int i = 0; i < dataGridView.Rows.Count; i++)
     {
         for (int j = 0; j < dataGridView.Columns.Count; j++)
         {
             worksheet.Cell(i + 2, j + 1).Value =
dataGridView.Rows[i].Cells[j].Value?.ToString();
         }
     }
     // Guardar el archivo de Excel
     SaveFileDialog saveFileDialog = new SaveFileDialog();
     saveFileDialog.Filter = "Archivos Excel (*.xlsx)|*.xlsx";
     saveFileDialog.FileName = "ArchivoExcel.xlsx";
     if (saveFileDialog.ShowDialog() == DialogResult.OK)
     {
         try
         {
             workbook.SaveAs(saveFileDialog.FileName);
             MessageBox.Show("El archivo Excel se ha exportado correctamente.",
"Exportar a Excel", MessageBoxButtons.OK, MessageBoxIcon.Information);

             // Abrir el archivo de Excel después de guardarlo
             System.Diagnostics.Process.Start(saveFileDialog.FileName);
         }
         catch (Exception ex)
         {
             MessageBox.Show("Error al guardar el archivo Excel: " + ex.Message,
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
         }
     }

    }
```

Este es un breve resumen de la programación de este módulo.

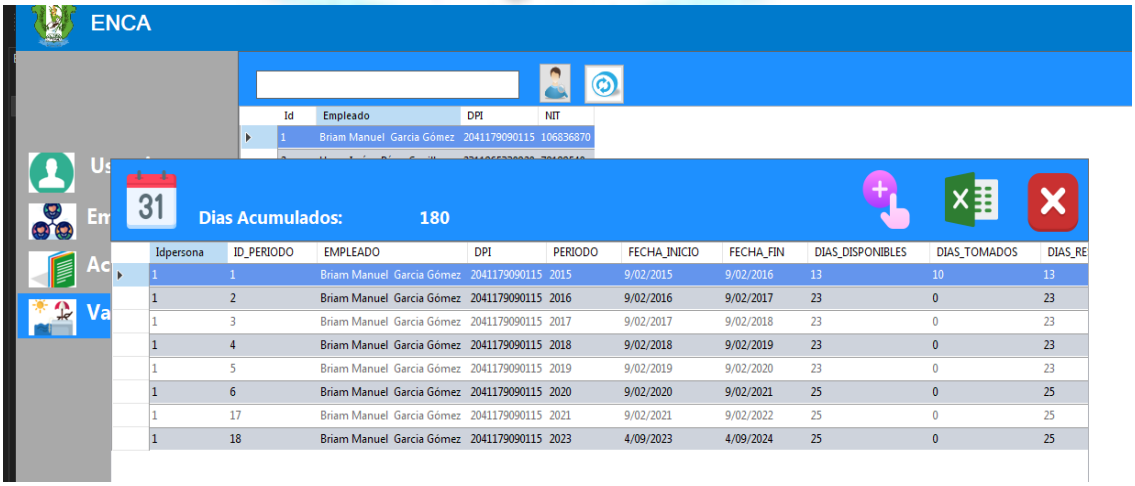
Asignación de vacaciones a empleados.

En esta parte se debe de agregar las vacaciones a los empleados en base a sus periodos que se le asignaron o tiene derecho el empleado, mediante una ventana la cual realiza ciertos cálculos matemáticos que son programados al sistema para que realiza cada una de las instrucciones que se definen para esta asignación de vacaciones.

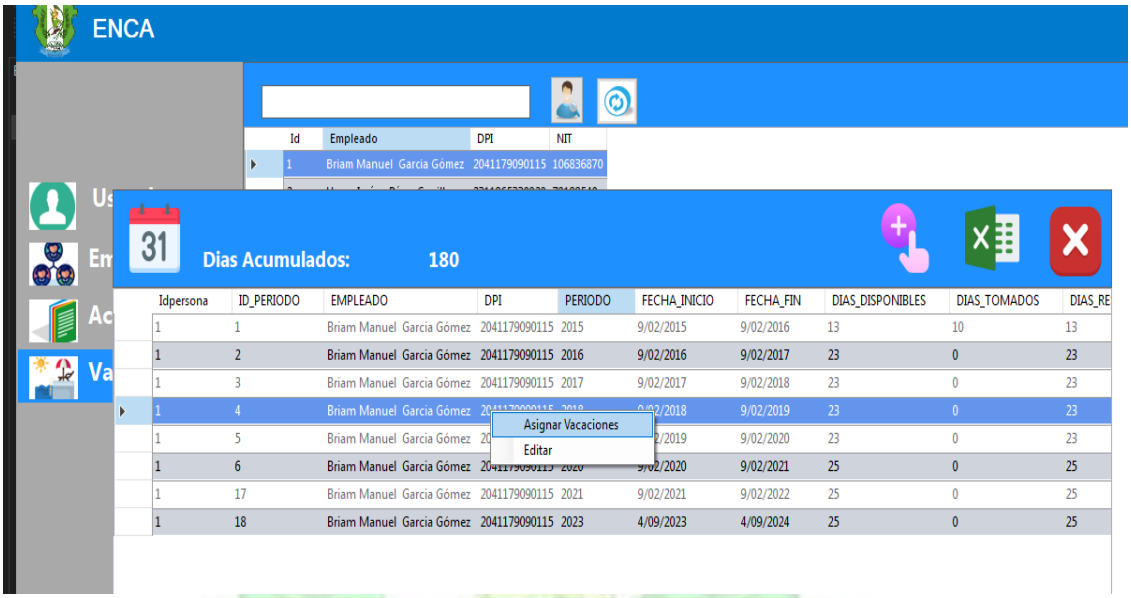
Para ello se define de la siguiente manera primero vamos al menú de vacaciones el cual tiene el registro de los empleados que se ingresan al sistema,



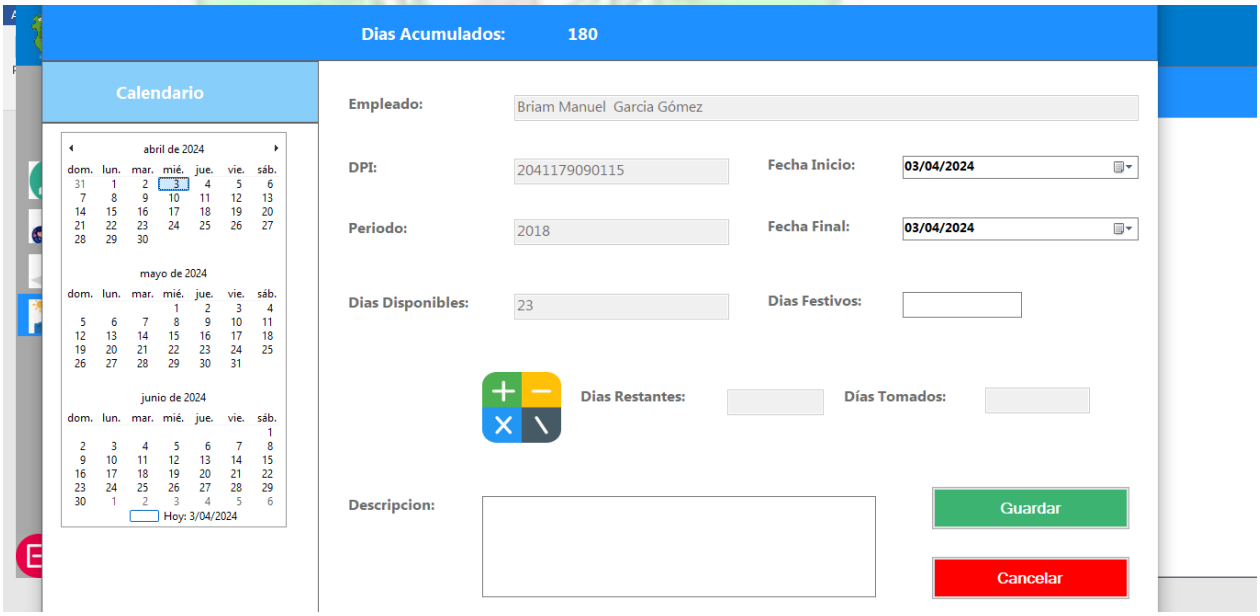
Aquí nos dirigimos al registro del empleado y al sub menú con clic derecho y seleccionamos agregar periodos.



En base a estos periodos de la persona se tiene que asignar las vacaciones del empleado para ello se debe de dar clic derecho sobre el periodo que se desea asignar vacaciones.

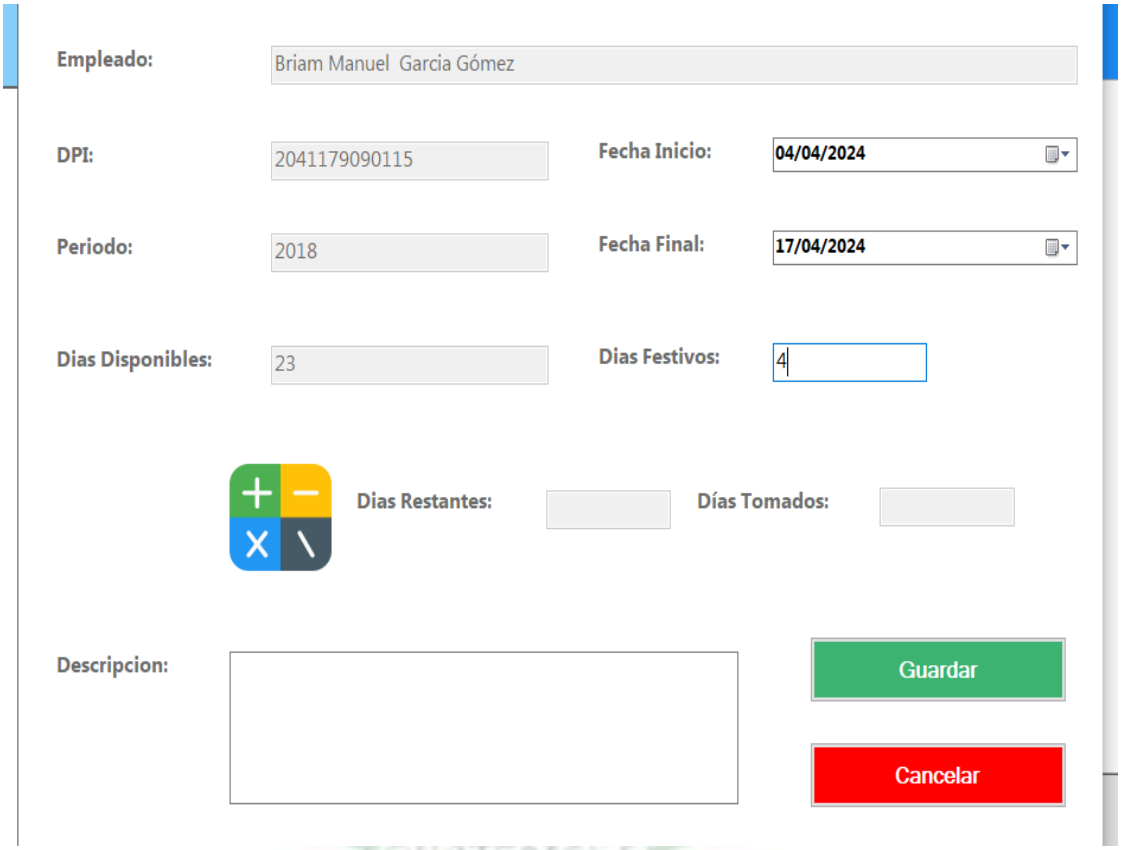


Damos clic.

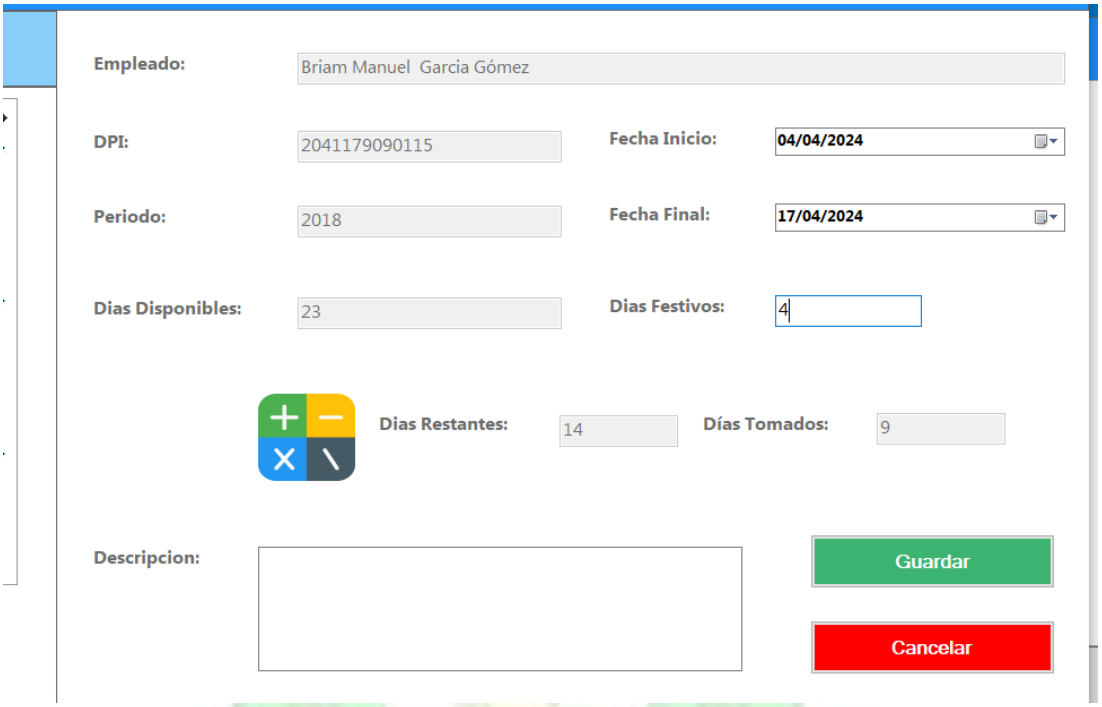


Nos despliega la siguiente pantalla.

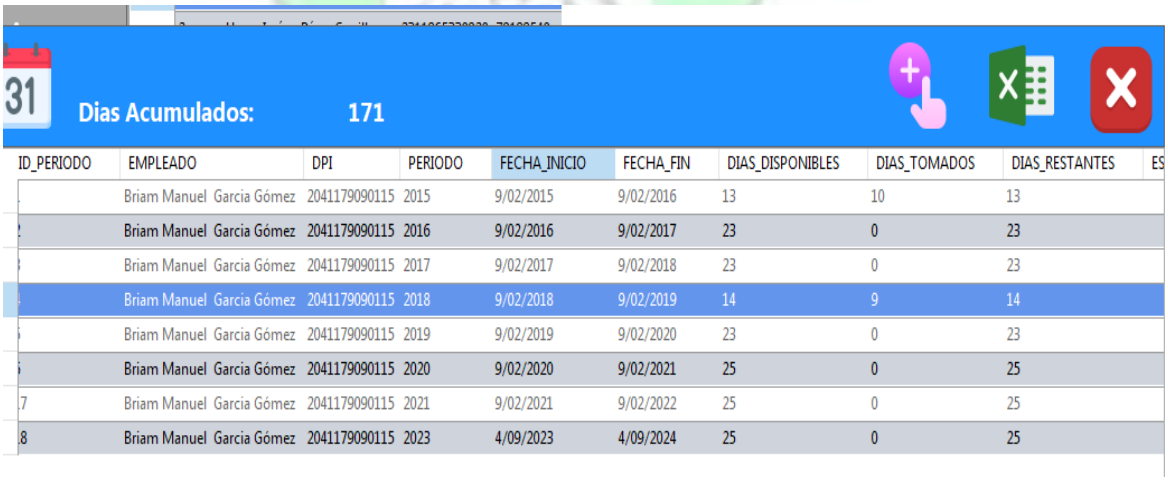
Aquí nos dice el nombre del empleado, DPI, el periodo que se está asignando las vacaciones, y los días disponibles que tiene de ese periodo, aquí la lógica es la siguiente en la parte donde dice Fecha de inicio y fecha de fin se debe de ingresar que de fecha a que fecha se van a asignar las vacaciones.



En este caso es del 4 al 17 de abril los días festivos son 4 entre esas fechas, en el botón que tiene los símbolos operativos básicos damos clic, este botón hace el cálculo según la programación



Y nos dice los días restantes son tantos y los días tomados tantos anulando los días festivos. Guardamos y nos actualiza el periodo con sus días tomados y días restantes.



ID_PERIODO	EMPLEADO	DPI	PERIODO	FECHA_INICIO	FECHA_FIN	DIAS_DISPONIBLES	DIAS_TOMADOS	DIAS_RESTANTES	ES
	Briam Manuel Garcia Gómez	2041179090115	2015	9/02/2015	9/02/2016	13	10	13	
	Briam Manuel Garcia Gómez	2041179090115	2016	9/02/2016	9/02/2017	23	0	23	
	Briam Manuel Garcia Gómez	2041179090115	2017	9/02/2017	9/02/2018	23	0	23	
	Briam Manuel Garcia Gómez	2041179090115	2018	9/02/2018	9/02/2019	14	9	14	
	Briam Manuel Garcia Gómez	2041179090115	2019	9/02/2019	9/02/2020	23	0	23	
	Briam Manuel Garcia Gómez	2041179090115	2020	9/02/2020	9/02/2021	25	0	25	
7	Briam Manuel Garcia Gómez	2041179090115	2021	9/02/2021	9/02/2022	25	0	25	
8	Briam Manuel Garcia Gómez	2041179090115	2023	4/09/2023	4/09/2024	25	0	25	

Como podemos ver.

Programación de asignación de vacaciones.

```
public partial class frmSolicitudVacaciones : Form
{
   public int IdPeriodo {  get; set; } //ID_PERIODO
   public int IdPersona {  get; set; }
    public string Evento {  get; set; }
    public int diasTomados { get; set; }
    public int Acumulados { get; set; }
    public frmSolicitudVacaciones()
    {
        InitializeComponent();

        dtmfechaInicio.Format = DateTimePickerFormat.Custom;
        dtmfechaInicio.CustomFormat = "dd/MM/yyyy";

        dtmfechaFinal.Format = DateTimePickerFormat.Custom;
        dtmfechaFinal.CustomFormat = "dd/MM/yyyy";
    }

    private void frmSolicitudVacaciones_Load(object sender, EventArgs e)
    {

    }

    public void CargarDatos( int dias,string Evento,string empleado, string
dpi,string periodo ,string diasDisponibles)
    {
        if (Evento == "Completo")
        {

            txtEmpleado.Text = empleado;
            txtDPI.Text = dpi;
            txtPeriodo.Text = periodo;
            txtDiasDisponibles.Text = diasDisponibles.ToString();
            lbAcomulados.Text = Acumulados.ToString();
        }
        else if(Evento == "Incompleto")
        {
            txtEmpleado.Text = empleado;
            txtDPI.Text = dpi;
            txtPeriodo.Text = periodo;

            // Calcular cuántos días proporcionalmente te corresponderían
            double proporcion = (double)dias / 365;
            int diasVacaciones = (int)Math.Round(23 * proporcion);
            int sub = (diasVacaciones - diasTomados);
            txtDiasDisponibles.Text = sub.ToString();
            lbAcomulados.Text = Acumulados.ToString();
        }
    }

    private void calculo()
    {
        int diasFestivos = Convert.ToInt32(txtDiasFestivos.Text);
```

```
        // Obtener las fechas seleccionadas de los DateTimePicker
        DateTime fechaInicio = dtmfechaInicio.Value;
        DateTime fechaFin = dtmfechaFinal.Value;
        // Calcular la diferencia en días entre las fechas
        TimeSpan diferencia = fechaFin - fechaInicio;
        int diasDiferencia = (int)diferencia.TotalDays;


        if(diasDiferencia > diasFestivos && diasDiferencia !=0){
            int suma = (diasDiferencia - diasFestivos);
            int diasDisponibles = Convert.ToInt32(txtDiasDisponibles.Text);
            if (diasDisponibles >= suma && diasDisponibles != 0)
            {
                int restantes = (diasDisponibles - suma);
                this.txtTotal.Text = suma.ToString();
                this.txtRestantes.Text = restantes.ToString();
            }
            else
            {
                MessageBox.Show("No puedes asignar mas dias de vacaciones de los que
tienes disponibles en este periodo");
            }

        }
        else
        {
            MessageBox.Show("Los dias feriados sobrepasan las fechas de
vacaciones");
        }




    }

    private void pictureBox1_Click(object sender, EventArgs e)
    {
        if(this.txtDiasFestivos.Text !="" )
        {
            calculo();
        }
        else
        {
            MessageBox.Show("Necesita ingresar todos los campos");
        }


    }

    private void button1_Click(object sender, EventArgs e)
    {
        this.Close();
    }
    private void MensajeOk(string mensaje)
    {
        MessageBox.Show(mensaje, "Sistema de Empleados", MessageBoxButtons.OK,
MessageBoxIcon.Information);
    }
```

```
    private void MensajeError(string mensaje)
    {
        MessageBox.Show(mensaje, "Sistema de Empleados", MessageBoxButtons.OK,
MessageBoxIcon.Error);
    }
    private void button2_Click(object sender, EventArgs e)
    {
        DateTime fechaInicio = dtmfechaInicio.Value;
        DateTime fechaFin = dtmfechaFinal.Value;

        if (string.IsNullOrEmpty(this.txtDiasFestivos.Text) ||
             string.IsNullOrEmpty(this.txtDescripcion.Text)||
            string.IsNullOrEmpty(this.dtmfechaInicio.Text) ||
            string.IsNullOrEmpty(this.dtmfechaFinal.Text)
             )
        {
            MensajeError("Falta ingresar algunos datos Remarcados");

        }
        else
        {

            string rpta = "";
            if (this.Evento == "Nuevo")
            {
                int disponibles, total;
                total = Convert.ToInt32(this.txtTotal.Text);
                disponibles = Convert.ToInt32(this.txtDiasDisponibles.Text);
                if (total <= disponibles)
                {
                    rpta = nVacaciones.InsertarVacaciones(
                    IdPersona,
                    IdPeriodo,
                    fechaInicio,
                    fechaFin,
                    Convert.ToInt32(this.txtDiasFestivos.Text),
                    Convert.ToInt32(this.txtTotal.Text),
                    this.txtDescripcion.Text
                   );
                }
                else
                {
                    MessageBox.Show("No puedes tomas mas dias de los disponibles en
este periodo");
                }

            }
            else if (this.Evento == "Editar")
            {

            }
            if (rpta.Equals("OK"))
            {
                int subtotal = (diasTomados + Convert.ToInt32(txtTotal.Text));
                string rptaP = "";
                this.MensajeOk(this.Evento == "Nuevo" ? "Se insertó de forma
correcta el registro" : "Se actualizó de forma correcta el registro");
```

```
            rptaP = nVacaciones.DescontarPeriodo(
                IdPeriodo,
                Convert.ToInt32(txtRestantes.Text),
                subtotal,
                Convert.ToInt32(txtRestantes.Text)

                );

            this.Close();
        }
        else
        {
            this.MensajeError(rpta);
        }
    }
   }
```

En el siguiente informe se estará finalizando la aplicación con los reportes que faltan y la parte de poder cancelar las vacaciones a los empleados si así se es necesario, y la instalación del servidor para poder ejecutar la aplicación ya para su uso de pruebas y posteriormente para uso de producción.

Conclusión.

En conclusión, el diseño de pantallas en Windows Form en C# para sistemas que involucren la gestión de usuarios, la generación de reportes y la administración de vacaciones es fundamental para garantizar una experiencia de usuario efectiva y satisfactoria. Al enfocarse en la claridad, la usabilidad y la funcionalidad, se puede crear una interfaz intuitiva que facilite la navegación y la interacción del usuario con el sistema. Además, es importante desarrollar funcionalidades robustas que permitan la gestión eficiente de los datos y la generación de informes precisos. En conjunto, un diseño bien pensado y una implementación cuidadosa de las funcionalidades son clave para el éxito de cualquier aplicación en Windows Form en C#.

Referencias.

Softwares de desarrollo.

https://www.microsoft.com/es-es/sql-server/sql-server-downloads

https://visualstudio.microsoft.com/es/vs/community/

https://learn.microsoft.com/en-us/answers/questions/1455410/how-to-show-design-a-new-report-in-report-viewer-t

Imágenes.

https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.tutorialesprogramacionya.com%2Fcsharpya%2Fdetalleconcepto.php%3Fcodigo%3D155%26inicio%3D20&psig=AOvVaw0AlUuszsAWJ9HWEWL1w7Z0&ust=1712155221339000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCPCexrLho4UDFQAAAAAdAAAAABAE

https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.umg.edu.gt%2F&psig=AOvVaw399rWkbxp24PamwDVm8ieA&ust=1712155397429000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCNimzYPio4UDFQAAAAAdAAAAABAE