

```
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;
using System.Runtime.InteropServices;
using Entities;
using BussinessLayer;

namespace Gym
{
    public partial class MainForm : Form
    {
        #region Instancias

        //Clases internas
        MetodosGenerales _metodosGenerales;

        //Capa de Negocio
        private readonly BussinessPersonas _bussinessPersonas;
        private readonly BussinessRegistrosLogs _bussinesRegistrosLogs;

        //Entidades
        private Personas _personas;
        private readonly Entities.Empleados _empleados;
        private readonly Entities.Registros_Logs _registrosLogs;

        #endregion

        #region Variables
        private bool sesionJefeOn;
        private int personaLogueada;
        private int idRegistroLogin;

        #endregion

        #region Load
        public MainForm(bool mainJefe, int idPersonaLogin, int idLastLogin)
        {
            InitializeComponent();

            //inicializo los componentes del formulario,
            //pero también todas las instancias de cada clase
            //que necesito utilizar a lo largo del form.
            _metodosGenerales = new MetodosGenerales();
            _personas = new Personas();
            _empleados = new Entities.Empleados();
            _bussinessPersonas = new BussinessPersonas();
            _bussinesRegistrosLogs = new BussinessRegistrosLogs();
            _registrosLogs = new Registros_Logs();

            //Estos argumentos que recibe el form, son los
            //que me dicen si el jefe es quien se logueó
            //y el Id de la persona que se logueó, para
            //poder utilizarlo luego.
            personaLogueada = idPersonaLogin;
        }
    }
}
```

```

idRegistroLogin = idLastLogin;
sesionJefeOn = mainJefe;
if (sesionJefeOn)
{
    btnConfiguracion.Visible = true;
    btnEmpleados.Visible = true;
}
else
{
    btnConfiguracion.Visible = false;
    btnEmpleados.Visible = false;
}
}

private void MainJefe_Load(object sender, EventArgs e)
{
    Login frm = new Login();
    frm.Close();
    AsignarNombre();
}

#endregion

#region Metodos encapsulados

private void BtnLogout_Click(object sender, EventArgs e)
{
    //Registramos el logout en la bdd
    _registrosLogs.Empleado_ID = personaLogueada;
    _registrosLogs.Fecha_LogOut = DateTime.Now;
    _registrosLogs.Registro_Log_ID = idRegistroLogin;
    _bussinesRegistrosLogs.RegistrarLogOut(_registrosLogs);

    //deslogueamos la sesión abierta
    Login frm = new Login();
    this.Hide();
    frm.Show();
}
private void AsignarNombre()
{
    _metodosGenerales.GetPersona();
    lblBienvenido.Text = "Bienvenido " + _metodosGenerales.nombrePersona.ToString() +
    "!";
}

private void CallOffForms(object Hijo)
{
    if (this.windowboxMain.Controls.Count > 0)
    {
        this.windowboxMain.Controls.RemoveAt(0);
    }

    Form formularioHijo = Hijo as Form;
    formularioHijo.TopLevel = false;
    this.windowboxMain.Controls.Add(formularioHijo);
    this.windowboxMain.Tag = formularioHijo;
    formularioHijo.Show();
}

#endregion

```

```
#region Foco Menu

private void FocusAsistencia()
{
    focoAsistencia.Visible = true;
    focoRegistro.Visible = false;
    focoPagos.Visible = false;
    focoPlanes.Visible = false;
    focoCaja.Visible = false;
    focoEmpleados.Visible = false;
}

private void FocusRegistro()
{
    focoAsistencia.Visible = false;
    focoRegistro.Visible = true;
    focoPagos.Visible = false;
    focoPlanes.Visible = false;
    focoCaja.Visible = false;
    focoEmpleados.Visible = false;
}

private void FocusPagos()
{
    focoAsistencia.Visible = false;
    focoRegistro.Visible = false;
    focoPagos.Visible = true;
    focoPlanes.Visible = false;
    focoCaja.Visible = false;
    focoEmpleados.Visible = false;
}

private void FocusPlanes()
{
    focoAsistencia.Visible = false;
    focoRegistro.Visible = false;
    focoPagos.Visible = false;
    focoPlanes.Visible = true;
    focoCaja.Visible = false;
    focoEmpleados.Visible = false;
}

private void FocusCaja()
{
    focoAsistencia.Visible = false;
    focoRegistro.Visible = false;
    focoPagos.Visible = false;
    focoPlanes.Visible = false;
    focoCaja.Visible = true;
    focoEmpleados.Visible = false;
}

private void FocusEmpleados()
{
    focoAsistencia.Visible = false;
    focoRegistro.Visible = false;
    focoPagos.Visible = false;
    focoPlanes.Visible = false;
    focoCaja.Visible = false;
    focoEmpleados.Visible = true;
}

#endregion

#region Cerrar y minimizar
private void btnClose_Click(object sender, EventArgs e)
{
```

```

//Registramos el logout en la bdd
_registroLogs.Empleado_ID = personaLogueada;
_registroLogs.Fecha_LogOut = DateTime.Now;
_registroLogs.Registro_Log_ID = idRegistroLogin;
_bussinesRegistrosLogs.RegistrarLogOut(_registrosLogs);
Application.Exit();
}

private void btnMinimize_Click(object sender, EventArgs e)
{
    this.WindowState = FormWindowState.Minimized;
}

#endregion

#region Arrastrar Form

/* No he logrado encapsularlo en una clase separada*/

//Se importan archivos DLL para poder realizar la captura de la acción del mouse
//y poder arrastrar los formularios desde donde se realiza la captura

[DllImport("user32.dll", EntryPoint = "ReleaseCapture")]
public extern static void ReleaseCapture();

[DllImport("user32.dll", EntryPoint = "SendMessage")]
public extern static void SendMessage(System.IntPtr hWnd, int wMsg, int wParam, int lParam);

private void TitleboxMain_MouseDown(object sender, MouseEventArgs e)
{
    ReleaseCapture();
    SendMessage(this.Handle, 0x112, 0xf012, 0);
}

#endregion

#region Menu

private void BtnAsistencia_Click(object sender, EventArgs e)
{
    CallOfForms(new Asistencias(personaLogueada));
    FocusAsistencia();
}

private void BtnRegistro_Click(object sender, EventArgs e)
{
    CallOfForms(new Clientes(personaLogueada));
    FocusRegistro();
}

private void BtnPagos_Click(object sender, EventArgs e)
{
    CallOfForms(new Pagos(personaLogueada));
    FocusPagos();
}

private void BtnPlanes_Click(object sender, EventArgs e)
{
    CallOfForms(new Planes(personaLogueada));
    FocusPlanes();
}


```

```
private void BtnCaja_Click(object sender, EventArgs e)
{
    CallOfForms(new Cajas(personaLogueada));
    FocusCaja();
}

private void BtnEmpleados_Click(object sender, EventArgs e)
{
    CallOfForms(new Empleados(personaLogueada));
    FocusEmpleados();
}

private void btnConfiguracion_Click(object sender, EventArgs e)
{
    CallOfForms(new Configuracion(personaLogueada));
    FocusAsistencia();
}

#endregion
}
```

```
using BussinessLayer;
using Entities;
using System;
using System.Drawing;
using System.Windows.Forms;

namespace Gym
{
    public partial class Login : Form
    {
        #region Instancias
        //Capa de negocio
        private readonly BussinessEmpleados _bussinessEmplados;
        private readonly BussinessRegistrosLogs _bussinessRegistrosLogs;

        //Capa de entidades
        private readonly Entities.Empleados _empleados;
        private readonly Tipos_Empleados _tiposEmpleados;
        private readonly Personas _personas = new Personas();
        private readonly Entities.Registros_Logs _registros_Logs;

        //Clases internas de la capa
        private readonly MetodosGenerales _metodosGenerales;

        #endregion

        #region Variables

        private string clave = string.Empty;
        private int idEmpleadoLogin;
        private int idLastLogin;
        private bool mainJefe;
        private bool primerLogueo = false;

        #endregion

        #region Load
        public Login()
        {
            InitializeComponent();
            _bussinessEmplados = new BussinessEmpleados();
            _empleados = new Entities.Empleados();
            _tiposEmpleados = new Tipos_Empleados();
            _metodosGenerales = new MetodosGenerales();
            _bussinessRegistrosLogs = new BussinessRegistrosLogs();
            _registros_Logs = new Registros_Logs();
        }

        private void Login_Load(object sender, EventArgs e)
        {
            btnIngresar.Focus();
            CerrarForms();
            VerificarPrimerLogin();
        }
    }

    #endregion

    #region Métodos encapsulados

    private void VerificarPrimerLogin()
    {
        bool resultado = _bussinessEmplados.ConsultarRegistrosLogin(primerLogueo);
    }
}
```

```
if (resultado)
{
    lblGenerarPrimerUsuario.Visible = true;
}
else
{
    lblGenerarPrimerUsuario.Visible = false;
}

private void CerrarForms()
{
    MainForm frm = new MainForm(mainJefe, idEmpleadoLogin, idLastLogin);
    frm.Close();
}

//Encriptamiento de clave
private void EncriptarClaveClave()
{
    //Encriptamos la clave con un HASH256
    //Y la asignamos al campo de la entidad correspondiente
    clave = EncriptamientoSHA256.GetSHA256(txtClave.Text.ToString());
}

private void RegistrarLogin()
{
    _registros_Logs.Empleado_ID = idEmpleadoLogin;
    _registros_Logs.Fecha_LogIn = DateTime.Now;
    _bussinessRegistrosLogs.RegistrarLogin(_registros_Logs);
    _registros_Logs.Empleado_ID = idEmpleadoLogin;
    _bussinessRegistrosLogs.GetLastLogID(_registros_Logs);
    idLastLogin = _registros_Logs.Registro_Log_ID;
}

#endregion

#region Eventos KeyPress

private void txtUsuario_Enter(object sender, EventArgs e)
{
    if (txtUsuario.Text == "Usuario")
    {
        txtUsuario.Text = string.Empty;
        txtUsuario.ForeColor = Color.Black;
    }
}
private void txtUsuario_Leave(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(txtUsuario.Text))
    {
        txtUsuario.Text = "Usuario";
        txtUsuario.ForeColor = Color.DimGray;
    }
}
private void txtClave_Enter(object sender, EventArgs e)
{
    if (txtClave.Text == "Clave")
    {
        txtClave.Text = string.Empty;
        txtClave.ForeColor = Color.Black;
    }
}
```

```
        }

    private void txtClave_Leave(object sender, EventArgs e)
    {
        if (string.IsNullOrEmpty(txtClave.Text))
        {
            txtClave.Text = "Clave";
            txtClave.ForeColor = Color.DimGray;
        }
    }

#endregion

#region Eventos Click

    private void lblGenerarPrimerUsuario_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
    {
        RegistroPrimeraVez registroPrimeraVez = new RegistroPrimeraVez();
        registroPrimeraVez.ShowDialog();
    }

    private void btnClose_Click(object sender, EventArgs e)
    {
        Application.Exit();
    }

    private void btnMinimize_Click(object sender, EventArgs e)
    {
        this.WindowState = FormWindowState.Minimized;
    }

    private void btnIngresar_Click(object sender, EventArgs e)
    {
        //Vamos a verificar primero si existe la clave en el sistema.
        //en caso de existir, entonces se va a abrir la pantalla
        //correspondiente si es Jefe o si es usuario

        EncriptarClaveClave();

        //asigno la clave a una variable, sin encriptarla (xq cuando traigo
        //la clave de la bdd, esta viene encriptada. Si yo encripto la clave
        //encriptada, la comparación siempre va a ser falsa)
        string usuario = txtUsuario.Text.ToString();

        _empleados.Usuario = txtUsuario.Text.ToString();
        _empleados.Clave = clave;
        _bussinessEmplados.VerificarClaveEnBdd(_tiposEmpleados, _empleados);

        //una vez traídos los datos, los comparamos con los que tenemos en
        //cada variable. Si son idénticas, entonces, ya puede verificarse
        //el acceso al programa
        if (_empleados.Clave == clave && _empleados.Usuario == usuario)
        {
            //Verificamos si es de un jefe o usuario,
            //para que el programa abra la parte correspondiente

            if (_tiposEmpleados.Estado == "Activo" && _tiposEmpleados.Acceso_Clave == "Y")
            {
                //Vamos a traer los datos del empleado que abrió sesión
                idEmpleadoLogin = _empleados.Empleado_ID;
```

```

//Si mainJefe es true, entonces se habilitan todos los botones
//del formulario principal que se va a abrir a continuación,
//sino, hay un botón que no se habilita xq no tienen los permisos
//correspondientes, cada uno de los usuarios del programa.
if (_tiposEmpleados.Tipo == "Jefe")
{
    mainJefe = true;
}
else
{
    mainJefe = false;
}
RegistrarLogin();
MainForm mj = new MainForm(mainJefe, idEmpleadoLogin, idLastLogin);
mj.Show();
this.Hide();
}
else
{
    //Cualquier otro estado no tiene la autorización correspondiente
    //vacaciones, carpeta médica)
    //Para acceder deberá solicitar permiso al jefe, y el mismo deberá cambiar
    el estado del empleado
    MessageBox.Show("Su estado actual no le permite acceder al programa. " +
                    "Por favor, comuníquese con su administrador " +
                    "para que haga la modificación de manera manual",
    "Advertencia",
    MessageBoxButtons.OK, MessageBoxIcon.Warning);
}
else
{
    MessageBox.Show("No hay usuario ni clave para los datos registrados. " +
                    "En caso de no ser usuario, deberá solicitar un perfil con el
    administrador correspondiente", "Error",
    MessageBoxButtons.OK, MessageBoxIcon.Error);
}

private void btnIngresar_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == Convert.ToChar(Keys.Enter))
    {
        btnIngresar_Click(sender, e);
    }
}

private void txtClave_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == Convert.ToChar(Keys.Enter))
    {
        btnIngresar_Click(sender, e);
    }
}

#endif

//MessageBox.Show("Se encontró el perfil", "Prueba");
//MessageBox.Show("No se encontró el perfil", "Prueba fallida");
}
}

```

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

namespace Gym
{
    public partial class RegistroPrimeraVez : Form
    {
        #region Instancias
        //entidades
        private Entities.Empleados _empleados;
        private readonly Personas _personas;

        //clases internas
        private readonly Restricciones _restricciones;
        private readonly MetodosGenerales _metodosGenerales;

        //Negocio
        private readonly BussinessEmpleados _bussinessEmpleados;
        private readonly BussinessPersonas _bussinessPersonas;
        #endregion

        #region Load
        public RegistroPrimeraVez()
        {
            InitializeComponent();
            _restricciones = new Restricciones();
            _metodosGenerales = new MetodosGenerales();
            _bussinessEmpleados = new BussinessEmpleados();
            _empleados = new Entities.Empleados();
            _bussinessPersonas = new BussinessPersonas();
            _personas = new Personas();
        }
        private void RegistroPrimeraVez_Load(object sender, EventArgs e)
        {
            Tipos_Documentos();
            Tipos_Sexos();
        }
        #endregion

        #region Variables

        private bool camposVacíos = true;
        private string clave = string.Empty;

        #endregion

        #region Métodos encapsulados

        private void VerificarCamposVacios()

    }
```

```

{
    //Recorro todos los textbox para saber si están vacíos
    //Y si hay alguno vacío, lo notifico, porque no pueden
    //haber campos vacíos en este formulario.
    foreach (Control txt in this.Controls)
    {
        if (txt is TextBox box)
        {
            TextBox t;
            t = box;

            //verificamos si algún txt está vacío
            if (string.IsNullOrEmpty(t.Text))
            {
                camposVacios = true;
                break;
            }
            else
            {
                camposVacios = false;
            }
        }
    }

    private void AltaPersona()
    {
        //Pasamos los datos a la entidad Personas
        _personas.Nombre = txtNombre.Text.ToString();
        _personas.Apellido = txtApellido.Text.ToString();
        _personas.Tipo_Documento_ID = cmbTipoDocumento.SelectedIndex;
        _personas.Nro_documento = txtNroDocumento.Text.ToString();
        _personas.Tipo_Sexo_ID = cmbSexo.SelectedIndex;
        _personas.Nro_Teléfono = txtTeléfono.Text.ToString();
        _personas.Fecha_Alta = DateTime.Now;
        _personas.Mail = txtMail.Text.ToString();

        //Y la enviamos a la capa de negocio para llevarla a la bdd.
        _bussinessPersonas.AltaJefe(_personas);
    }

    private void AltaEmpleado()
    {
        //Primero obtenemos el último ID de la persona registrada
        _metodosGenerales.GetLastID();
        //luego ese ID lo asignamos a la entidad Empleados
        _empleados.Persona_ID = _metodosGenerales.persona_ID;
        _empleados.Tipo_Emppleado_ID = 1;
        _empleados.Estado_Emppleado_ID = 0;
        _empleados.Usuario = txtUsuario.Text.ToString();

        //Encriptamos la clave con un HASH256
        clave = Convert.ToString(txtClave.Text);
        //Y la asignamos al campo de la entidad correspondiente
        _empleados.Clave = EncriptamientoSHA256.GetSHA256(clave);

        //Envío toda la data a la capa de negocio para ser mandada a la bdd.
        _bussinessEmpleados.AltaEmpleado(_empleados);
    }
}

```

```

private void Tipos_Documentos()
{
    //llamo al método para traer los tipos de documentos y los asigno al combobox
    _metodosGenerales.Bring_Tipos_Documentos();
    cmbTipoDocumento.DataSource = _metodosGenerales.DtTipos_Documentos;
    cmbTipoDocumento.DisplayMember = "Tipo";
    cmbTipoDocumento.ValueMember = "Tipo_documento_ID";
}

private void Tipos_Sexos()
{
    //llamo al método para traer los tipos de sexos y los asigno al combobox
    _metodosGenerales.Bring_Tipos_Sexos();
    cmbSexo.DataSource = _metodosGenerales.DtTipos_Sexos;
    cmbSexo.DisplayMember = "Sexo"; //Pero solo esta columna es la que muestro
    cmbSexo.ValueMember = "Tipo_Sexo_ID";
}

#endregion

#region Eventos KeyPress

//Tengo un método encapsulado en la clase Restricciones para los campos
//que quiero que sean numéricos.
//Así que creo una variable, le asigno la tecla presionada
//y si no es numérico, lo bloquea.
//Es una clase general, xq todos los controles de los forms, que son numéricos
//utilizan el mismo método.
//Para no crear nuevamente el método en cada formulario.
private void txtNroDocumento_KeyPress(object sender, KeyPressEventArgs e)
{
    string strTexto = txtNroDocumento.Text;
    _restricciones.SoloNumeros(e, strTexto);
}

private void txtTelefono_KeyPress(object sender, KeyPressEventArgs e)
{
    string strTexto = txtTelefono.Text;
    _restricciones.SoloNumeros(e, strTexto);
}

#endregion

#region Eventos Botones

private void lblCancelar_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
    VerificarCamposVacios();
    if (!camposVacios)
    {
        DialogResult cancelar = MessageBox.Show("Faltan campos sin rellenar. ¿Seguro
que quiere cancelar la operación? s" +
presione Cancelar. Caso contrario, presione Aceptar",
                                            "En caso de seguir con el formulario,
                                            "Datos incompletos",
                                            MessageBoxButtons.OKCancel);
        if (cancelar == DialogResult.OK)
        {
            this.Close();
        }
    }
    else
    {
}
}

```

```
        this.Close();
    }
}

private void btnAceptar_Click(object sender, EventArgs e)
{
    //Verificamos campos vacíos
    VerificarCamposVacios();
    if (camposVacios)
    {
        //Se solicita que llene todos los campos
        MessageBox.Show("Para poder dar de alta el registro, deben completarse todos los campos.",
                        "Campos incompletos",
                        MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
    else
    {
        //Se da el alta del empleado
        AltaPersona();
        AltaEmpleado();
        DialogResult result = MessageBox.Show("Se generó su perfil de manera correcta.  
Desde ahora puede ingresar al sistema con su usuario y clave.", "Registro exitoso",
                                                MessageBoxButtons.OK, MessageBoxIcon.Information);
        if (result == DialogResult.OK)
        {
            this.Close();
        }
    }
}

#endregion

}
```

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

namespace Gym
{
    public partial class Asistencias : Form
    {
        #region Instancias
        //Capa de negocio
        private readonly BussinessClientes _bussinesClientes;
        private readonly BussinessPlanes _bussinessPlanes;
        private readonly BussinessAsistencia _bussinessAsistencia;
        private readonly BussinessPlanesAsignados _bussinesPlanesAsignados;

        //Entities
        private readonly Entities.Clientes _clientes;
        private readonly Entities.Planes _planes;
        private readonly Entities.Asistencias _asistencias;
        private readonly Entities.Planes_Asignados _planesAsignados;

        //clases internas
        private readonly Restricciones _restricciones;

        #endregion

        #region Variables

        private int idEmpleadoLogin;
        private string buscar;
        private DataSet DsClienteAsistencia;
        private DataSet DsAsistencias;
        private DataTable DtPlanes;
        private DataSet DsPlanesAsignados;
        private DataTable dtAlumnosTotales;
        private DataTable dtAlumnosPresentes;
        private bool camposVacios = true;
        private int cliente_ID;
        private int planSeleccionado;
        private string diaDeLaSemana;
        private string fechaBusqueda;
        private bool listadoCargado = false;

        #endregion

        #region Loading
        public Asistencias(int idPersonaLog)
        {
            InitializeComponent();
            idEmpleadoLogin = idPersonaLog;

            _bussinessPlanes = new BussinessPlanes();
            _bussinesClientes = new BussinessClientes();
        }
    }
}
```

```

_bussinessAsistencia = new BussinessAsistencia();
_bussinesPlanesAsignados = new BussinessPlanesAsignados();

_planes = new Entities.Planes();
_clientes = new Entities.Clientes();
_asistencias = new Entities.Asistencias();
_planesAsignados = new Entities.Planes_Asignados();

_restricciones = new Restricciones();

BuscarPlanes();
buscarAsistenciasDiarias();

diaDeLaSemana = GetDiaDeLaSemana(DateTime.Now.ToString());
BuscarPlanesParaAsistenciaAlumnos();

btnAsignarPlan.Enabled = false;
}

//Si no busco un cliente en particular, debería poder tmb cargar los planes del día
//y buscar el listado de alumnos por plan.

#endregion

#region Métodos encapsulados
private void GetJornadaDePlan()
{
    bool esEmpleado = false;
    bool darBaja = false;
    int plan_ID = planSeleccionado;
    Jornadas jn = new Jornadas(plan_ID, esEmpleado, darBaja);
    jn.ShowDialog();
}

private void GetAlumnosDeLaClase()
{
    //Vamos a convertir la fecha en el formato que corresponde.
    DateTime fechaPresente = dtFechabusqueda.Value;
    //fechaPresente = fechaPresente.Substring(0, 10);

    //alumnos totales
    listAlumnosAusentes.DisplayMember = string.Empty;
    listAlumnosPresentes.DisplayMember = string.Empty;

    //Necesito traer solo los ausentes
    _planes.Plan_ID = Convert.ToInt32(cmbClasesParaAsistencia.SelectedValue);
    dtAlumnosTotales = _bussinessPlanes.GetAlumnoPorClase(_planes, fechaPresente);

    listAlumnosAusentes.DataSource = dtAlumnosTotales;
    listAlumnosAusentes.DisplayMember = "NombreCliente";
    listAlumnosAusentes.ValueMember = "Cliente_ID";

    //Alumnos presentes en la clase por fecha
    _planes.Plan_ID = Convert.ToInt32(cmbClasesParaAsistencia.SelectedValue);
    dtAlumnosPresentes = _bussinessPlanes.GetAlumnoPresentes(_planes, fechaPresente);

    listAlumnosPresentes.DataSource = dtAlumnosPresentes;
    listAlumnosPresentes.DisplayMember = "NombreCliente";
    listAlumnosPresentes.ValueMember = "Cliente_ID";
}

```

```
private void buscarAsistenciasDiarias()
{
    if (txtBuscarAsistencias.Text == "Buscar")
    {
        buscar = string.Empty;
    }
    else
    {
        buscar = txtBuscarAsistencias.Text;
    }
    DsAsistencias = _bussinessAsistencia.GetAsistenciasDiarias(buscar);

    if (DsAsistencias.Tables[0].Rows.Count > 0)
    {
        foreach (DataRow dr in DsAsistencias.Tables[0].Rows)
        {
            switch (dr[5].ToString())
            {
                case "A":
                    dtgvAsistencias.Rows.Add(dr[0].ToString(), dr[1], dr[2], dr[3],
dr[4], "Ausente");
                    break;
                case "P":
                    dtgvAsistencias.Rows.Add(dr[0].ToString(), dr[1], dr[2], dr[3],
dr[4], "Presente");
                    break;
                default:
                    dtgvAsistencias.Rows.Add(dr[0].ToString(), dr[1], dr[2], dr[3],
dr[4], "Tarde");
                    break;
            }
        }
    }
}

private void BuscarPlanes()
{
    DtPlanes = _bussinessPlanes.GetPlanes(_planes);
    cmbPlanesActivos.DataSource = DtPlanes;
    cmbPlanesActivos.DisplayMember = "Nombre";
    cmbPlanesActivos.ValueMember = "Plan_ID";
}

private void BusquedaDeClientesById()
{
    DsClienteAsistencia =
_bussinesClientes.BuscarClienteAsistenciaById(_planesAsignados);
    ResetControlsCliente();
    AcomodarDatos();
    if (camposVacios)
    {
        camposVacios = false;
    }
}

private void BuscarDatosCliente()
{
    DsClienteAsistencia = _bussinesClientes.BuscarClienteAsistencia(buscar);
    AcomodarDatos();
    DsPlanesAsignados = _bussinesPlanesAsignados.BuscarPlanesAsignados(cliente_ID);
    AcomodarPlanesAsignados();
}
```

```

private void BuscarPlanesParaAsistenciaAlumnos()
{
    diaDeLaSemana = Normalizar(diaDeLaSemana);
    _planes.Estado = "A";
    DtPlanes = _bussinessPlanes.GetPlanesParaAsistencia(_planes, diaDeLaSemana);
    cmbClasesParaAsistencia.DataSource = DtPlanes;
    cmbClasesParaAsistencia.DisplayMember = "Nombre";
    cmbClasesParaAsistencia.ValueMember = "Plan_ID";
}

private string Normalizar(string strDato)
{
    /*se crean dos variables que van a contener las vocales con tilde y sin tilde
     para luego poder reemplazar las que tengan tilde, por las que no y
     poder hacer una busqueda más exhaustiva*/
    string strVocales = "aeiou";
    string strVocalesTilde = "áéíóú";

    //agarra la primer letra del parámetro de busqueda
    for (int i = 0; i < strDato.Length; i++)
    {
        //hace el recorrido sobre el length de las vocales.
        //como miden lo mismo, se puede poner 4. Sino, podría reemplazarse
        // por "strVocales.length - 1" que sería 4.
        //va 4 porque se cuenta el 0, contando el 0 no son 4 sino 5.
        for (int j = 0; j < 4; j++)
        {
            //la condición dice que si la letra que se está utilizando del parámetro
            de búsqueda
                //es igual a una de las vocales de las variables con tilde que se está
                recorriendo (con tilde
                    //porque recordemos que hay que reemplazar las que tienen tilde por las
                    que no)
                    //entonces va a entrar en este if.
                    if (strDato[i] == strVocalesTilde[j])
                    {
                        //para hacer que esa letra sea reemplazada por su vocal sin tilde.
                        strDato = strDato.Replace(strDato[i], strVocales[j]);

                        //rompe este último for, y pasa a la siguiente letra del parámetro de
                        búsqueda
                        break;
                    }
                }
            }

        //Colcaremos la primera letra del string, en mayúscula tmb.
        string letras = "lmjvs";
        string Letrotas = "LMJVS";
        //agarra la primer letra del parámetro de busqueda
        for (int i = 0; i < 1; i++)
        {
            //hace el recorrido sobre el length de las vocales.
            //como miden lo mismo, se puede poner 4. Sino, podría reemplazarse
            // por "strVocales.length - 1" que sería 4.
            //va 4 porque se cuenta el 0, contando el 0 no son 4 sino 5.
            for (int j = 0; j < 4; j++)
            {
                //la condición dice que si la letra que se está utilizando del parámetro
                de búsqueda
                    //es igual a una de las vocales de las variables con tilde que se está

```

```

    recorriendo (con tilde
                    //porque recordemos que hay que reemplazar las que tienen tilde por las
                    que no)
                    //entonces va a entrar en este if.
                    if (strData[i] == letras[j])
                    {
                        //para hacer que esa letra sea reemplazada por su vocal sin tilde.
                        strData = strData.Replace(strData[i], Letrotas[j]);
                    }

                    //rompe este último for, y pasa a la siguiente letra del parámetro de
                    búsqueda
                    break;
                }
            }

        //como el método es un string, debe devolver un string
        return strData;
    }

private void AcomodarDatos()
{
    if (DsClienteAsistencia.Tables[0].Rows.Count > 0)
    {
        foreach (DataRow dr in DsClienteAsistencia.Tables[0].Rows)
        {
            cliente_ID = int.Parse(dr[1].ToString());
            lblNombreCliente.Text += dr[2].ToString() + " " + dr[3].ToString();
            lblNro_documento.Text += dr[4].ToString();
            lblTelefono.Text += dr[5].ToString();
            lblMail.Text += dr[6].ToString();
            break;
        }
        camposVacios = false;
        btnAsignarPlan.Enabled = true;
    }
    else
    {
        MessageBox.Show("No hay clientes con el documento ingresado. " +
                       "Por favor, intente de nuevo, o haga primero el registro del cliente.",
                       "Datos no encontrados");
        camposVacios = true;
    }
}

private void AcomodarPlanesAsignados()
{
    int i = 0;
    foreach (DataRow dr in DsPlanesAsignados.Tables[0].Rows)
    {
        lblPlanActual.Text += dr[0].ToString();
        i++;
        if ((i + 1) <= DsPlanesAsignados.Tables[0].Rows.Count)
        {
            lblPlanActual.Text += ", ";
        }
    }
}

private void ColocarAsistencia()
{
    _asistencias.Cliente_ID = Convert.ToInt32(listAlumnosAusentes.SelectedValue);
    _asistencias.Fecha = DateTime.Now;
    _asistencias.Estado = "P";
}

```

```
_asistencias.Empleado_ID = idEmpleadoLogin;
_asistencias.Plan_Asignado_ID =
Convert.ToInt32(cmbClasesParaAsistencia.SelectedValue);
_bussinessAsistencia.PutAsistencia(_asistencias);
}
private void BuscarDatosPlan()
{
    ResetControlsPlanes();
    _planes.Plan_ID = Convert.ToInt32(cmbPlanesActivos.SelectedValue);
    _bussinessPlanes.GetDatosPlan(_planes);
    planSeleccionado = _planes.Plan_ID;
    lblCostoMensual.Text += _planes.Importe_Plan.ToString();
    if (_planes.Cupo_Total < 1)
    {
        lblCuposTotales.Text = "0";
        lblCuposRestantes.Text = "0";
    }
    else
    {
        lblCuposTotales.Text = _planes.Cupo_Total.ToString();
        lblCuposRestantes.Text = _planes.Cupo_Restante.ToString();
    }
}

private void ResetControlsPlanes()
{
    lblCostoMensual.Text = "Costo mensual: $";
    lblCuposRestantes.Text = "0";
    lblCuposTotales.Text = "0";
}

private void ResetControlsCliente()
{
    lblNombreCliente.Text = "Nombre: ";
    lblNro_documento.Text = "DNI: ";
    lblTelefono.Text = "Telefono: ";
    lblMail.Text = "Mail: ";
    lblPlanActual.Text = "Clases Actuales: ";
}

private void AsigarlePlanAlCliente()
{
    _planesAsignados.Plan_ID = Convert.ToInt32(cmbPlanesActivos.SelectedValue);
    _planesAsignados.Empleado_ID = idEmpleadoLogin;
    _planesAsignados.Cliente_ID = cliente_ID;
    _planesAsignados.Fecha_Inscripcion = DateTime.Now;
    _planesAsignados.Estado = "A";
    _bussinesPlanesAsignados.AsginarPlanAlCliente(_planesAsignados);

    _planes.Plan_ID = Convert.ToInt32(cmbPlanesActivos.SelectedValue);
    _bussinessPlanes.EditarCupoRestante(_planes);
}

private void BusquedaDeClientes()
{
    if (!string.IsNullOrEmpty(buscar))
    {
        ResetControlsCliente();
        BuscarDatosCliente();
        buscar = string.Empty;
    }
}
```

```

        else
    {
        MessageBox.Show("Tiene que buscar con un número de documento válido. No se
admienten campos vacíos.",
                    "Gestión en proceso", MessageBoxButtons.OKCancel);
    }
}

#endregion

#region Eventos
private void cmbPlanesActivos_SelectionChangeCommitted(object sender, EventArgs e)
{
    //Me busca el precio, cupo total y restante.
    BuscarDatosPlan();
    //Busca los horarios
}
private void lblVerJornadas_LinkClicked(object sender, LinkLabelLinkClickedEventArgs
e)
{
    GetJornadaDePlan();
}
private void cmbClasesParaAsistencia_SelectionChangeCommitted(object sender, EventArgs
e)
{
    //verificar el día de la semana y buscar por día
    //debo obtener qué día de la semana es ya sea para una búsqueda
    //para ahora, o para otro día del pasado.
    //sabiendo el día, puedo buscar determinado día en las jornadas
    //y ya para búsquedas del pasado, solamente debo saber la fecha y buscar
    //en las asistencias y hacer las comparaciones correspondientes.

    diaDeLaSemana = GetDiaDeLaSemana(dtFechabusqueda.Value.ToString());
    GetAlumnosDeLaClase();
    listadoCargado = true;
}

private string GetDiaDeLaSemana(string fecha)
{
    //separo por partes la fecha
    int dia = Convert.ToInt32(fecha.Substring(0, 2));
    int mes = Convert.ToInt32(fecha.Substring(3, 2));
    int anio = Convert.ToInt32(fecha.Substring(6, 4));

    //paso los parametros a una nueva variable (no pude hacerlo
    //directo desde el datetimepicker)
    DateTime dateValue = new DateTime(anio, mes, dia);

    //y devuelvo en string el día de la semana en español
    return dateValue.ToString("dddd", new CultureInfo("es-ES"));
}

private void listAlumnos_DoubleClick(object sender, EventArgs e)
{
    _planesAsignados.Cliente_ID = Convert.ToInt32(listAlumnosAusentes.SelectedValue);
    _planesAsignados.Plan_ID = Convert.ToInt32(cmbClasesParaAsistencia.SelectedValue);

    if (camposVacios)
    {
        BusquedaDeClientesById();
        camposVacios = false;
    }
}

```

```

    else
    {
        DialogResult result = MessageBox.Show("Tiene un cliente cargado. ¿Quiere
interrumpir la gestión y buscar un cliente nuevo?",
                                            "Gestión en proceso", MessageBoxButtons.OKCancel);
        if (result == DialogResult.OK)
        {
            BusquedaDeClientesById();
            camposVacios = false;
        }
    }

//if (cliente_ID != _planesAsignados.Cliente_ID)
//{
//
//}

}

private void txtBuscarCliente_KeyPress(object sender, KeyPressEventArgs e)
{
    listAlumnosAusentes.ClearSelected();
    buscar = txtBuscarCliente.Text;
    _restricciones.SoloNumeros(e, buscar);
    if (e.KeyChar == Convert.ToChar(Keys.Enter))
    {
        if (camposVacios)
        {
            BusquedaDeClientes();
        }
        else
        {
            DialogResult result = MessageBox.Show("Tiene un cliente cargado. ¿Quiere
interrumpir la gestión y buscar un cliente nuevo?",
                                            "Gestión en proceso", MessageBoxButtons.OKCancel);
            if (result == DialogResult.OK)
            {
                BusquedaDeClientes();
            }
        }
    }
}

private void LlamadoALaAsistencia()
{
    ColocarAsistencia();
    buscarAsistenciasDiarias();
    if (lblNombreCliente.Text == "Nombre: ")
    {
        string nombreAlumno = listAlumnosAusentes.SelectedItem.ToString();
        MessageBox.Show($"Asistencia registrada correctamente para {nombreAlumno}, el
día de la fecha.",
                      "Asistencia OK.", MessageBoxButtons.OKCancel);
    }
    else
    {
        string nombre = lblNombreCliente.Text;
        nombre = nombre.Substring(8, nombre.Length - 8);
        MessageBox.Show($"Asistencia registrada correctamente para
{nombre.ToString()}, el día de la fecha.",
                      "Asistencia OK.", MessageBoxButtons.OKCancel);
    }
    GetAlumnosDeLaClase();
}

```

```
        }

    private void btnGuardarAsistencia_Click(object sender, EventArgs e)
    {
        if (camposVacios)
        {
            DialogResult result = MessageBox.Show("No hay alumno cargado, pero si el
listado. ¿Desea colocarle asistencia al alumno seleccionado?", 
                                         "Asistencia", MessageBoxButtons.OKCancel);
            if (result == DialogResult.OK)
            {
                LlamadoALaAsistencia();
            }
        }
        else
        {
            LlamadoALaAsistencia();
        }
    }

private void btnAsignarPlan_Click(object sender, EventArgs e)
{
    if (camposVacios)
    {
        MessageBox.Show("No hay cliente seleccionado para asignarle un plan." +
                      "Por favor, busque el cliente primero, y luego asigne el plan.");
    }
    else
    {
        if (Convert.ToInt32(lblCuposRestantes.Text) == 0)
        {
            MessageBox.Show("No hay cupos disponibles para esta clase.");
        }
        else
        {
            //Asigarle el plan.
            AsigarlePlanAlCliente();
        }
    }
}

#endregion

#region Focus Textbox
private void txtBuscarCliente_Enter(object sender, EventArgs e)
{
    if (txtBuscarCliente.Text == "DNI")
    {
        txtBuscarCliente.Text = string.Empty;
        txtBuscarCliente.ForeColor = Color.Black;
    }
}

private void txtBuscarCliente_Leave(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(txtBuscarCliente.Text))
    {
        txtBuscarCliente.Text = "DNI";
        txtBuscarCliente.ForeColor = Color.DimGray;
    }
}

private void txtBuscarAsistencias_Enter(object sender, EventArgs e)
```

```
{  
    if (txtBuscarAsistencias.Text == "Buscar")  
    {  
        txtBuscarAsistencias.Text = string.Empty;  
        txtBuscarAsistencias.ForeColor = Color.Black;  
    }  
}  
  
private void txtBuscarAsistencias_Leave(object sender, EventArgs e)  
{  
    if (string.IsNullOrEmpty(txtBuscarAsistencias.Text))  
    {  
        txtBuscarAsistencias.Text = "Buscar";  
        txtBuscarAsistencias.ForeColor = Color.DimGray;  
    }  
}  
  
#endregion  
  
private void BuscarPlanesAsistenchaConFecha()  
{  
    diaDeLaSemana = Normalizar(diaDeLaSemana);  
    _planes.Esto  
    DtPlanes = _bussinessPlanes.GetPlanesConFecha(_planes, diaDeLaSemana,  
fechaBusqueda);  
    cmbClasesParaAsistencia.DataSource = DtPlanes;  
    cmbClasesParaAsistencia.DisplayMember = "Nombre";  
    cmbClasesParaAsistencia.ValueMember = "Plan_ID";  
}  
  
private void dtFechabusqueda_ValueChanged(object sender, EventArgs e)  
{  
    //diaDeLaSemana = GetDiaDeLaSemana(dtFechabusqueda.Value.ToString());  
    //fechaBusqueda = dtFechabusqueda.Value.ToString();  
    //BuscarPlanesAsistenchaConFecha();  
}  
}  
}
```

```
using BussinessLayer;
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;
using Entities;

namespace Gym
{
    public partial class Cajas : Form
    {
        #region Instancias
        //Clases internas
        private readonly MetodosGenerales _metodosGenerales;
        private readonly Restricciones _restricciones;

        //Capa negocio
        private readonly BussinessCaja _bussinessCaja;

        //Entidades
        private readonly Entities.Cajas _caja;
        private readonly Entities.Detalles_Cajas _detalles_Cajas;
        private readonly SaldosActualizados _saldos_Actualizados;

        #endregion

        #region Variables
        private int personaLogueada;
        private decimal ingreso;
        private decimal egreso;
        private decimal importeFinalCaja = -1;
        private DataSet DsCajas;
        private int idCajaAbierta;
        private string buscar;
        private bool cajaAbierta;
        private bool primeraCaja;

        #endregion

        public Cajas(int idPersonaLog)
        {
            InitializeComponent();
            _restricciones = new Restricciones();
            _metodosGenerales = new MetodosGenerales();
            _bussinessCaja = new BussinessCaja();
            _caja = new Entities.Cajas();
            _detalles_Cajas = new Entities.Detalles_Cajas();
            _saldos_Actualizados = new SaldosActualizados();
            personaLogueada = idPersonaLog;
            VerificarCajaAbierta();
            ActualizacionDeImportes();
            GetDetallesCajas();
        }
    }
}
```

```

#region Encapsulamientos

private void VerificarCajaAbierta()
{
    DateTime fecha = DateTime.Now;
    cajaAbierta = _bussinessCaja.VerificarCajaAbierta(fecha);
    if (cajaAbierta)
    {
        gbAperturaCaja.Enabled = false;
        gbCierreCaja.Enabled = true;
        GetLastCajaID();
    }
    else
    {
        gbAperturaCaja.Enabled = true;
        gbCierreCaja.Enabled = false;
    }
}

private void GetLastCajaID()
{
    _bussinessCaja.GetLastCajaID(_caja);
    idCajaAbierta = _caja.Caja_ID;
}

private void ActualizacionDeImportes()
{
    //Falta tener en cuenta el importe de ingreso que se registra en caja
    //cuando se abre la caja.
    _saldos_Actualizados.Caja_ID = idCajaAbierta;
    _bussinessCaja.ConsultarSaldos(_saldos_Actualizados);

    lblImporteInicial.Text = _saldos_Actualizados.Importe_Inicial.ToString();
    lblIngresos.Text = _saldos_Actualizados.Importe_Ingreso.ToString();
    lblEgresos.Text = _saldos_Actualizados.Importe_Egreso.ToString();
    lblTotal.Text = _saldos_Actualizados.Total.ToString();
}

private void GetDetallesCajas()
{
    DsCajas = _bussinessCaja.GetDetallesCajas(buscar);

    dtgvCajas.Rows.Clear();
    //Vacío el textbox para resetear en caso de búsqueda específica
    buscar = string.Empty;

    //Y acá traigo los datos
    //Pregunto si el dataset tiene datos, preguntando si tiene filas
    if (DsCajas.Tables[0].Rows.Count > 0)
    {
        //Y por cada fila que haya en el dataset
        foreach (DataRow dr in DsCajas.Tables[0].Rows)
        {
            string fecha = dr[1].ToString();
            fecha = fecha.Substring(0, fecha.Length - 8);

            dtgvCajas.Rows.Add(dr[0].ToString(), fecha, dr[2], dr[3], dr[4], dr[5]);
        }
    }
}

```

```

private void VerificarImportes(KeyPressEventArgs e, string _importe)
{
    string importe = _importe.ToString();
    _restricciones.SoloNumeros(e, importe);
}
private bool VerificarBoxes()
{
    bool estaVacio;

    if (string.IsNullOrEmpty(txtImporteFinal.Text) ||
Convert.ToInt32(txtImporteFinal.Text) < 0)
    {
        estaVacio = true;
    }
    else
    {
        estaVacio = false;
    }

    return estaVacio;
}
private void VerificarImportesFinales()
{
    DialogResult result = MessageBox.Show($"El importe final es de {importeFinalCaja}.\n¿Es correcto? De ser correcto, presione \"Aceptar\", " +
        $"de lo contrario presione \"Cancelar\" y reingrese el importe final",
    "Verificar Importe de Cierre de Caja");

    if (result == DialogResult.OK)
    {
        CerrarCaja();
    }
}

private void CerrarCaja()
{
    _caja.Caja_ID = idCajaAbierta;
    _caja.Empleado_ID_Cierre = personaLogueada;
    _caja.Fecha_Cierre = DateTime.Now;
    _caja.Importe_Cierre = Convert.ToDecimal(lblTotal.Text);
    _caja.Importe_Cierre_Caja = importeFinalCaja;
    _caja.Caja_Abierta = false;

    _bussinessCaja.CerrarCaja(_caja);

}
#endregion

private void btnAbrirCaja_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(txtImporteEfectivo.Text))
    {
        MessageBox.Show("Tiene que ingresar el importe de inicio. Si la caja está
vacía, " +
                        "ingrese un 0.", "Campos vacíos!");
    }
    else
    {
        _caja.Importe_Aertura = Convert.ToDecimal(txtImporteEfectivo.Text);
        _caja.Empleado_ID_Aertura = personaLogueada;
    }
}

```

```
_caja.Fecha_Aertura = DateTime.Now;
_caja.Caja_Abierta = true;

//Abro la caja del día
_bussinessCaja.AbrirCaja(_caja);
cajaAbierta = true;

//consultamos el último ID
GetLastCajaID();

//Actualizamos los datos para que se muestren los detalles de cada día.
GetDetallesCajas();

gbAperturaCaja.Enabled = false;
gbCierreCaja.Enabled = true;
}

}

private void txtImporteEfectivo_KeyPress(object sender, KeyPressEventArgs e)
{
    string importe = txtImporteEfectivo.Text;
    VerificarImportes(e, importe);
}

private void txtBuscarCajas_KeyPress(object sender, KeyPressEventArgs e)
{
    buscar = txtBuscarCajas.Text;
    if (e.KeyChar == (char)Keys.Enter)
    {
        GetDetallesCajas();
    }
}

private void txtImporteFinal_KeyPress(object sender, KeyPressEventArgs e)
{
    string importe = txtImporteFinal.Text;
    ActualizacionDeImportes();
    VerificarImportes(e, importe);

    if (e.KeyChar == (char)Keys.Enter)
    {
        importeFinalCaja = Convert.ToDecimal(importe);
        lblImporteCajaFinal.Text = txtImporteFinal.Text;
        lblDiferencia.Text =
Convert.ToString(Convert.ToDecimal(lblImporteCajaFinal.Text) -
Convert.ToDecimal(lblTotal.Text));
    }
}

private void btnCerrar_Click(object sender, EventArgs e)
{
    bool estaVacio = VerificarBoxes();
    if (estaVacio)
    {
        MessageBox.Show("Primero debe ingresar el importe de cierre en el cuadro de texto correspondiente.", "Importe final faltante");
    }
    else
    {
        importeFinalCaja = Convert.ToDecimal(txtImporteFinal.Text);
        decimal total = Convert.ToDecimal(lblTotal.Text);
```

```
decimal suma = importeFinalCaja - total;

DialogResult result = MessageBox.Show(
    $"Verifique si los montos son correctos:\n" +
    $"Importe en Caja: ${importeFinalCaja}.\n" +
    $"Importe registrado en el sistema: ${total}.\n" +
    $"Diferencia: {suma}.\n" +
    "Si estos importes son correctos, seleccione 'Aceptar', de lo contrario
elija 'Cancelar'." +
    "Recuerde que estos importes no pueden modificarse una vez que elija
'Aceptar'.",
    "Leer bien - Cierre de Caja",
    MessageBoxButtons.OKCancel,
    MessageBoxIcon.Exclamation);
if (result == DialogResult.OK)
{
    //Cerramos la caja.
    CerrarCaja();
    VerificarCajaAbierta();
    ActualizacionDeImportes();
    GetDetallesCajas();
    MessageBox.Show("Ha cerrado la caja con éxito!", "Cierre de Caja",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
}
}
}
}
```

```
using BussinessLayer;
using Entities;
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 Gym
{
    public partial class Clientes : Form
    {
        #region Instancias
        //Entidades
        private readonly Entities.Clientes _clientes;
        private readonly Personas _personas;

        //Clases internas
        private readonly Restricciones _restricciones;
        private readonly MetodosGenerales _metodosGenerales;

        //Capa de negocio
        private readonly BussinessClientes _bussinessClientes;
        private readonly BussinessPersonas _bussinessPersonas;

        #endregion

        #region Variables
        private int personaLogueada;

        private bool contenidoErroneo;
        private bool camposObligatoriosVacios;
        //Verificar coincidencias ****
        private bool personaRegistrada;
        private DataSet dsTablaClientes;
        private string buscar;
        private int motivoMenu = 5;
        private bool edicionCliente = false;

        #endregion

        #region Load del formulario
        public Clientes(int idPersonalLog)
        {
            InitializeComponent();
            personaLogueada = idPersonalLog;
            _clientes = new Entities.Clientes();
            _restricciones = new Restricciones();
            _metodosGenerales = new MetodosGenerales();
            _bussinessClientes = new BussinessClientes();
            _bussinessPersonas = new BussinessPersonas();
            _personas = new Personas();
        }
        private void Clientes_Load(object sender, EventArgs e)
        {
            Tipos_Documentos();
            Tipos_Sexos();
        }
    }
}
```

```

        GetClientes();
    }

#endregion

#region Métodos Encapsulados

private void GetClientes()
{
    //Primero limpio el grid. Para que no haya errores.
    //Sería como si hiciera un refresh.
    //Saco lo que haya y luego vuelvo a traer los datos
    dtgvCliente.Rows.Clear();

    //Acá traigo los datos y se los asigno a la tabla dsTablaClientes
    dsTablaClientes = _bussinessClientes.GetClientes(buscar);
    //Vacío el textbox para resetear en caso de búsqueda específica
    buscar = string.Empty;

    //Y acá traigo los datos
    //Pregunto si el dataset tiene datos, preguntando si tiene filas
    if (dsTablaClientes.Tables[0].Rows.Count > 0)
    {
        //Y por cada fila que haya en el dataset
        foreach (DataRow dr in dsTablaClientes.Tables[0].Rows)
        {
            if (dr[6].ToString() == "A")
            {
                //paso esa fila al datagrid para que se vean los datos
                dtgvCliente.Rows.Add(dr[0].ToString(), dr[1], dr[2], dr[3], dr[4],
dr[5], "Activo");
            }
            else
            {
                dtgvCliente.Rows.Add(dr[0].ToString(), dr[1], dr[2], dr[3], dr[4],
dr[5], "Inactivo");
            }
        }
    }
}

private void Tipos_Documentos()
{
    //llamo al método para traer los tipos de documentos y los asigno al combobox
    _metodosGenerales.Bring_Tipos_Documentos();
    cmbTipoDocumentoCliente.DataSource = _metodosGenerales.DtTipos_Documentos;
    //lo que quiero ver
    cmbTipoDocumentoCliente.DisplayMember = "Tipo";
    //y les asigno el id correspondiente
    cmbTipoDocumentoCliente.ValueMember = "Tipo_documento_ID";
}

private void Tipos_Sexos()
{
    //llamo al método para traer los tipos de sexos y los asigno al combobox
    _metodosGenerales.Bring_Tipos_Sexos();
    cmbSexo.DataSource = _metodosGenerales.DtTipos_Sexos;
    cmbSexo.DisplayMember = "Sexo"; //Pero solo esta columna es la que muestro
    cmbSexo.ValueMember = "Tipo_Sexo_ID";
}

private void ABMCliente()

```

```
{  
    if (motivoMenu == 3)  
    {  
        EditarPersona();  
        EditarCliente();  
    }  
    else  
    {  
        if (motivoMenu == 5)  
        {  
            AltaPersona();  
            AltaCliente();  
        }  
    }  
}  
  
private void EditarPersona()  
{  
    //Es la edición de los datos personales  
    //junto con los datos de empleado, excepto las claves  
    int Persona_ID = _personas.Persona_ID;  
    string Nombre = Convert.ToString(txtNombreCliente.Text);  
    string Apellido = Convert.ToString(txtApellidoCliente.Text);  
    int Tipo_Documento_ID = cmbTipoDocumentoCliente.SelectedIndex;  
    string Nro_documento = Convert.ToString(txtNumDocumentoCliente.Text);  
    int Tipo_Sexo_ID = cmbSexo.SelectedIndex;  
    string Nro_Telefono = Convert.ToString(txtTelefonoCliente.Text);  
    string Nro_Alternativo, Mail, Observaciones;  
  
    //Envío todo el objeto creado a la cada de negocios para ser enviado  
    //a la bdd.  
  
    //Acá verifico si el usuario ingresó datos nuevos o no a los campos  
    //que reciben nulos.  
    //Si no agregaron datos, los mandamos como nulos  
    //caso contrario, se envía lo que haya en su correspondiente Textbox  
    if (txtAlternativoCliente.Text != "Alternativo")  
    {  
        Nro_Alternativo = Convert.ToString(txtAlternativoCliente.Text);  
    }  
    else  
    {  
        Nro_Alternativo = string.Empty;  
    }  
    if (txtMailCliente.Text != "Mail")  
    {  
        Mail = Convert.ToString(txtMailCliente.Text);  
    }  
    else  
    {  
        Mail = string.Empty;  
    }  
    if (txtObservacionesCliente.Text != "Observaciones y/o consideraciones")  
    {  
        Observaciones = Convert.ToString(txtObservacionesCliente.Text);  
    }  
    else  
    {  
        Observaciones = string.Empty;  
    }  
    _metodosGenerales.EditarPersona(Persona_ID, Nombre,  
}
```

```

        Apellido,
        Tipo_Documento_ID,
        Nro_documento,
        Tipo_Sexo_ID,
        Nro_Telefono,
        Nro_Alternativo,
        Mail,
        Observaciones);
    }

private void EditarCliente()
{
    if (cmbEstado.SelectedIndex == 0)
    {
        _clientes.Estado = "A";
    }
    else
    {
        _clientes.Estado = "I";
    }
    _bussinessClientes.EditarCliente(_clientes);
}

private void AltaPersona()
{
    //En la clase "MetodosGenerales" tengo un método para enviar los datos de
    //una nueva persona a la entidad y luego a la base de datos a través
    //de la capa de negocio
    string Nombre = Convert.ToString(txtNombreCliente.Text);
    string Apellido = Convert.ToString(txtApellidoCliente.Text); ;
    int Tipo_Documento_ID = cmbTipoDocumentoCliente.SelectedIndex;
    string Nro_Documento = Convert.ToString(txtNumDocumentoCliente.Text);
    int Tipo_Sexo_ID = cmbSexo.SelectedIndex;
    string Nro_Telefono = Convert.ToString(txtTelefonoCliente.Text);
    string Nro_Alternativo;
    string Mail;
    string Observaciones;
    DateTime FechaAlta = DateTime.Now;

    if (txtAlternativoCliente.Text != "Alternativo")
    {
        Nro_Alternativo = Convert.ToString(txtAlternativoCliente.Text);
    }
    else
    {
        Nro_Alternativo = string.Empty;
    }

    if (txtMailCliente.Text != "Mail")
    {
        Mail = Convert.ToString(txtMailCliente.Text);
    }
    else
    {
        Mail = string.Empty;
    }

    if (txtObservacionesCliente.Text != "Observaciones y/o consideraciones")
    {
        Observaciones = Convert.ToString(txtObservacionesCliente.Text);
    }
}

```

```

    else
    {
        Observaciones = string.Empty;
    }

    //Así que asigno cada campo a una variable y mando esas variables como argumentos
    //para el método en la clase.
    //Eso lo hago para poder utilizar el mismo método desde otro formulario
    _metodosGenerales.AltaPersona(Nombre,
                                    Apellido,
                                    Tipo_Documento_ID,
                                    Nro_Documento,
                                    Tipo_Sexo_ID,
                                    Nro_Teléfono,
                                    Nro_Alternativo,
                                    Mail,
                                    Observaciones,
                                    FechaAlta);

}

private void AltaCliente()
{
    //El ID de la última persona que se registró (método anterior)
    //Tenía otra sentencia, para traer el ID que se generó
    //En esta primera línea, traigo el valor de esta variable y o asigno.
    _clientes.Persona_ID = _metodosGenerales.persona_ID;

    //Luego relleno los campos faltantes de la entidad Cliente.
    if (cmbEstado.SelectedIndex == 0 || cmbEstado.SelectedItem == null)
    {
        _clientes.Estado = "A";
    }
    else
    {
        _clientes.Estado = "I";
    }
    _bussinessClientes.AltaCliente(_clientes);
}

private void ValidarCamposVacios()
{
    //Estos campos que valido, son los obligatorios.
    if (string.IsNullOrEmpty(txtNombreCliente.Text) ||
        string.IsNullOrEmpty(txtApellidoCliente.Text) ||
        string.IsNullOrEmpty(txtNumDocumentoCliente.Text) ||
        string.IsNullOrEmpty(txtTelefonoCliente.Text))
    {
        camposObligatoriosVacios = true;
    }
    else
    {
        camposObligatoriosVacios = false;
    }
}

private void ValidarContenido()
{
    //acá valido el contenido de los campos obligatorios
    //xq pueden estar llenos pero con el texto que los identifica como "Nombre"
    //Y esos strings son los que no quiero que se guarden en la base de datos
    foreach (Control txt in this.gbClientes.Controls)
}

```

```

    {
        if (txt is TextBox)
        {
            if (txt.Text == "Nombre" ||
                txt.Text == "Apellido" ||
                txt.Text == "Documento" ||
                txt.Text == "Teléfono")
            {
                MessageBox.Show("Existen campos con datos erróneos. Por favor, rellene todos los campos con datos válidos", "Campos obligatorios sin llenar.", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                contenidoErroneo = true;
                break;
            }
            else
            {
                contenidoErroneo = false;
            }
        }
    }

private void ResetControls()
{
    //Reseteamos todos los controles del form.
    txtNombreCliente.Text = "Nombre";
    txtApellidoCliente.Text = "Apellido";
    txtNumDocumentoCliente.Text = "Documento";
    txtTelefonoCliente.Text = "Teléfono";
    txtAlternativoCliente.Text = "Alternativo";
    txtMailCliente.Text = "Mail";
    txtObservacionesCliente.Text = "Observaciones y/o consideraciones";
    cmbTipoDocumentoCliente.SelectedItem = 0;
    cmbSexo.SelectedItem = 0;
    foreach (Control txt in this.gbClientes.Controls)
    {
        if (txt is TextBox box)
        {
            txt.ForeColor = Color.DimGray;
        }
    }
}

private void VerificarCoincidencias()
{
    if (motivoMenu != 3)
    {
        int id = _clientes.Persona_ID;
        string documento = txtNumDocumentoCliente.Text.ToString();
        _bussinessPersonas.BuscarCoincidencias(id, documento, _personas);
        if (_personas.Persona_ID == id && _personas.Nro_documento == documento)
        {
            personaRegistrada = true;
        }
        else
        {
            personaRegistrada = false;
        }
    }
}

#endregion

```

```
#region Eventos de Foco en TextBox

private void txtNombreCliente_Enter(object sender, System.EventArgs e)
{
    //Primero preguntamos si el texto dice Nombre
    //Al principio siempre lo va a decir, porque así
    //inicializa el form. Por lo que lo vamos a vaciar
    //y cambiar el color de la letra.
    //En caso de que no diga literalmente Nombre
    //Se va a dejar como está, ya que puede ser
    //un texto que el usuario escribió

    if (txtNombreCliente.Text == "Nombre")
    {
        txtNombreCliente.Text = string.Empty;
        txtNombreCliente.ForeColor = Color.Black;
    }
}

private void txtNombreCliente_Leave(object sender, System.EventArgs e)
{
    //Por el lado contrario, cuando se quita el foco
    //del control, lo volvemos a su estado y color inicial

    if (string.IsNullOrEmpty(txtNombreCliente.Text))
    {
        txtNombreCliente.Text = "Nombre";
        txtNombreCliente.ForeColor = Color.DimGray;
    }
}

/*Así igual para los siguientes eventos dentro de esta región*/
private void txtApellidoCliente_Enter(object sender, System.EventArgs e)
{
    if (txtApellidoCliente.Text == "Apellido")
    {
        txtApellidoCliente.Text = string.Empty;
        txtApellidoCliente.ForeColor = Color.Black;
    }
}

private void txtApellidoCliente_Leave(object sender, System.EventArgs e)
{
    if (string.IsNullOrEmpty(txtApellidoCliente.Text))
    {
        txtApellidoCliente.Text = "Apellido";
        txtApellidoCliente.ForeColor = Color.DimGray;
    }
}

private void txtNumDocumentoCliente_Enter(object sender, System.EventArgs e)
{
    if (txtNumDocumentoCliente.Text == "Nº documento")
    {
        txtNumDocumentoCliente.Text = string.Empty;
        txtNumDocumentoCliente.ForeColor = Color.Black;
    }
}

private void txtNumDocumentoCliente_Leave(object sender, System.EventArgs e)
{
    if (string.IsNullOrEmpty(txtNumDocumentoCliente.Text))
    {
```

```
        txtNumDocumentoCliente.Text = "Nº documento";
        txtNumDocumentoCliente.ForeColor = Color.DimGray;
    }
}
private void txtTelefonoCliente_Enter(object sender, System.EventArgs e)
{
    if (txtTelefonoCliente.Text == "Teléfono")
    {
        txtTelefonoCliente.Text = string.Empty;
        txtTelefonoCliente.ForeColor = Color.Black;
    }
}
private void txtTelefonoCliente_Leave(object sender, System.EventArgs e)
{
    if (string.IsNullOrEmpty(txtTelefonoCliente.Text))
    {
        txtTelefonoCliente.Text = "Teléfono";
        txtTelefonoCliente.ForeColor = Color.DimGray;
    }
}

private void txtAlternativoCliente_Enter(object sender, System.EventArgs e)
{
    if (txtAlternativoCliente.Text == "Alternativo")
    {
        txtAlternativoCliente.Text = string.Empty;
        txtAlternativoCliente.ForeColor = Color.Black;
    }
}

private void txtAlternativoCliente_Leave(object sender, System.EventArgs e)
{
    if (string.IsNullOrEmpty(txtAlternativoCliente.Text))
    {
        txtAlternativoCliente.Text = "Alternativo";
        txtAlternativoCliente.ForeColor = Color.DimGray;
    }
}

private void txtMailCliente_Enter(object sender, System.EventArgs e)
{
    if (txtMailCliente.Text == "Mail")
    {
        txtMailCliente.Text = string.Empty;
        txtMailCliente.ForeColor = Color.Black;
    }
}

private void txtMailCliente_Leave(object sender, System.EventArgs e)
{
    if (string.IsNullOrEmpty(txtMailCliente.Text))
    {
        txtMailCliente.Text = "Mail";
        txtMailCliente.ForeColor = Color.DimGray;
    }
}

private void txtObservacionesCliente_Enter(object sender, System.EventArgs e)
{
    if (txtObservacionesCliente.Text == "Observaciones y/o consideraciones")
    {
```

```

        txtObservacionesCliente.Text = string.Empty;
        txtObservacionesCliente.ForeColor = Color.Black;
    }

}

private void txtObservacionesCliente_Leave(object sender, System.EventArgs e)
{
    if (string.IsNullOrEmpty(txtObservacionesCliente.Text))
    {
        txtObservacionesCliente.Text = "Observaciones y/o consideraciones";
        txtObservacionesCliente.ForeColor = Color.DimGray;
    }
}
private void txtBuscarCliente_Enter(object sender, EventArgs e)
{
    if (txtBuscarCliente.Text == "Buscar")
    {
        txtBuscarCliente.Text = string.Empty;
        txtBuscarCliente.ForeColor = Color.Black;
    }
}

private void txtBuscarCliente_Leave(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(txtBuscarCliente.Text))
    {
        txtBuscarCliente.Text = "Buscar";
        txtBuscarCliente.ForeColor = Color.DimGray;
    }
}

#endregion

#region Eventos KeyPress en Textbox

private void txtNumDocumentoCliente_KeyPress(object sender, KeyPressEventArgs e)
{
    string strTexto = txtNumDocumentoCliente.Text;
    _restricciones.SoloNumeros(e, strTexto);
}

private void txtTelefonoCliente_KeyPress(object sender, KeyPressEventArgs e)
{
    string strTexto = txtTelefonoCliente.Text;
    _restricciones.SoloNumeros(e, strTexto);
}

private void txtAlternativoCliente_KeyPress(object sender, KeyPressEventArgs e)
{
    string strTexto = txtAlternativoCliente.Text;
    _restricciones.SoloNumeros(e, strTexto);
}

private void txtBuscarCliente_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == Convert.ToChar(Keys.Enter))
    {
        buscar = txtBuscarCliente.Text;
        GetClientes();
    }
}

```

```

#endregion

#region Alta Clientes

private void btnAltaCliente_Click(object sender, EventArgs e)
{
    //Verificamos si el contenido es distinto al texto predeterminado
    //Nombre, Apellido, etc. Tiene que ser distinto a eso,
    //para que sea válido
    ValidarContenido();
    if (!contenidoErroneo)
    {
        //Valido si hay campos obligatorios vacíos
        ValidarCamposVacios();
        if (!camposObligatoriosVacios)
        {
            VerificarCoincidencias();
            if (!personaRegistrada)
            {
                ABMCliente();
                ResetControls();
                GetClientes();
                motivoMenu = 5;
                edicionCliente = false;
            }
            else
            {
                //Vamos a mostrar un mensaje, en caso de que esta persona ya esté
                registrada.

                MessageBox.Show("Esta persona ya está registrada en el sistema. Solo
                se pueden hacer modificaciones", "Registro encontrada", MessageBoxButtons.OK,
                MessageBoxIcon.Information);
            }
        }
        else
        {
            MessageBox.Show("Campos obligatorios sin completar. Por favor, complete
            todos los campos requeridos", "Campos vacíos", MessageBoxButtons.OK);
        }
    }
}
#endregion

private void btnEditarDatosCliente_Click(object sender, EventArgs e)
{
    motivoMenu = 3;

    if (edicionCliente)
    {
        DialogResult result = MessageBox.Show("Tiene una edición en proceso. " +
            "En caso de querer cancelar la edición actual presione Aceptar", "Edición
            en proceso",
            MessageBoxButtons.OKCancel);
        if (result == DialogResult.OK)
        {
            CargarDatos();
        }
    }
    else
    {
        CargarDatos();
    }
}

```

```

        }

    private void CargarDatos()
    {
        //Asigno el ID de la fila, que está oculto
        //y lo paso como argumento para buscar los datos de ese id en la bdd
        _personas.Persona_ID = Convert.ToInt32(dtgvCliente.CurrentRow.Cells[1].Value);
        _bussinessPersonas.GetPersonaUnica(_personas);

        _clientes.Cliente_ID = Convert.ToInt32(dtgvCliente.CurrentRow.Cells[0].Value);
        _bussinessClientes.GetCliente(_clientes);

        txtNombreCliente.Text = _personas.Nombre;
        txtApellidoCliente.Text = _personas.Apellido;
        cmbTipoDocumentoCliente.SelectedIndex = _personas.Tipo_Documento_ID;
        txtNumDocumentoCliente.Text = _personas.Nro_documento;
        cmbSexo.SelectedIndex = _personas.Tipo_Sexo_ID;
        txtTelefonoCliente.Text = _personas.Nro_Telefono;
        if (string.IsNullOrEmpty(_personas.Nro_Alternativo))
        {
            txtAlternativoCliente.Text = "Alternativo";
        }
        else
        {
            txtAlternativoCliente.Text = _personas.Nro_Alternativo;
        }
        if (string.IsNullOrEmpty(_personas.Mail))
        {
            txtMailCliente.Text = "Mail";
        }
        else
        {
            txtMailCliente.Text = _personas.Mail;
        }
        if (string.IsNullOrEmpty(_personas.Observaciones))
        {
            txtObservacionesCliente.Text = "Observaciones y/o consideraciones";
        }
        else
        {
            txtObservacionesCliente.Text = _personas.Observaciones;
        }
        if (_clientes.Estado == "A")
        {
            cmbEstado.SelectedIndex = 0;
        }
        else
        {
            cmbEstado.SelectedIndex = 1;
        }

        foreach (Control txt in gbClientes.Controls)
        {
            if (txt is TextBox box)
            {
                TextBox t;
                t = box;
                if (t.ForeColor == Color.DimGray &&
                    t.Text != "Alternativo" &&
                    t.Text != "Mail" &&
                    t.Text != "Mail" &&
                    t.Text != "Observaciones y/o consideraciones")
                {
                    t.ForeColor = Color.Black;
                }
            }
        }
    }
}

```

```
        t.Text != "Observaciones y/o consideraciones")
    {
        t.ForeColor = Color.Black;
    }
}
edicionCliente = true;
}

private void btnEliminarCliente_Click(object sender, EventArgs e)
{
    _clientes.Cliente_ID = Convert.ToInt32(dtgvCliente.CurrentRow.Cells[0].Value);
    _clientes.Estado = "I";
    _bussinessClientes.BajaCliente(_clientes);
    GetClientes();
    motivoMenu = 5;
    edicionCliente = false;
}
}
```

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

namespace Gym
{
    public partial class Empleados : Form
    {
        #region Instancias
        //entidades
        private Entities.Empleados _empleados;
        private readonly Personas _personas;
        //realizar el registro de logueos
        private Registros_Logs _registroLogs;

        //clases internas
        private Restricciones _restricciones;
        private readonly MetodosGenerales _metodosGenerales;

        //Negocio
        private readonly BussinessEmpleados _bussinessEmpleados;
        private readonly BussinessPersonas _bussinessPersonas;
        private readonly BussinessJornadas _bussinessJornadas;

        #endregion

        #region variables
        private int personaLogueada;
        private bool camposObligatoriosVacios = true;
        private bool contenidoErroneo = false;
        private string clave = string.Empty;
        //Verificar coincidencias
        private bool personaRegistrada;
        private bool claveOK;
        private DataSet dsTablaEmpleados;
        private string buscar;
        private bool edicionEmpleado;
        private int motivoEdicion = 3;
        private bool chkTodos;
        private int persona_Jor;
        private bool cargarJornada;
        #endregion

        #region Load del form
        public Empleados(int idPersonaLog)
        {
            InitializeComponent();

            _restricciones = new Restricciones();
            _metodosGenerales = new MetodosGenerales();
        }
    }
}
```

```

_bussinessEmpleados = new BussinessEmpleados();
_bussinessPersonas = new BussinessPersonas();
_bussinessJornadas = new BussinessJornadas();

personaLogueada = idPersonaLog;

_empleados = new Entities.Empleados();
_personas = new Personas();
}

private void Empleados_Load(object sender, EventArgs e)
{
    Tipos_Documentos();
    Tipos_Sexos();
    Tipos_Empleados();
    Estados_Empleados();
    GetEmpleados();
}

#endregion

#region Métodos Encapsulados

private void CargarJornada()
{
    //Hacemos la solicitud para cargar el form de jornadas
    //enviándole el dato correspondiente para que cargue
    //las jornadas del empleado si es que hay, o directamente
    //haga la gestión que el usuario desee.
    bool empleado = true;
    bool darBaja;
    if (Convert.ToInt32(cmbEstados.SelectedValue) == 3)
    {
        darBaja = true;
    }
    else
    {
        darBaja = false;
    }
    Jornadas jn = new Jornadas(persona_Jor, empleado, darBaja);
    jn.ShowDialog();
}
private void BuscarEmpleado()
{
    //Asigno el ID de la fila, que está oculto
    //y le paso como argumento para buscar los datos de ese id en la bdd
    _personas.Persona_ID = Convert.ToInt32(dtgvEmpleados.CurrentRow.Cells[0].Value);
    _bussinessPersonas.GetPersonaUnica(_personas);

    _metodosGenerales.personaID = _personas.Persona_ID;
    _empleados.Persona_ID = _metodosGenerales.personaID;
    _bussinessEmpleados.GetTipoEmpleado(_empleados);
    _metodosGenerales.empleadoID = _empleados.Empleado_ID;

    //Luego de traer todos los datos, se los asigno a cada textbox

    //Datos de Tabla Personas
    txtNombreEmpleado.Text = _personas.Nombre;
    txtApellidoEmpleado.Text = _personas.Apellido;
    cmbTipoDocumentoEmpleado.SelectedIndex = _personas.Tipo_Documento_ID;
    txtDocumentoEmpleado.Text = _personas.Nro_documento;
    cmbSexoEmpleado.SelectedIndex = _personas.Tipo_Sexo_ID;
}

```

```

txtTelefonoEmpleado.Text = _personas.Nro_Teléfono;
txtAlternativoEmpleado.Text = _personas.Nro_Alternativo;
txtMailEmpleado.Text = _personas.Mail;
txtObservacionesEmpleado.Text = _personas.Observaciones;

//Datos de tabla Empleados
Tipos_Employados();
cmbTipoEmpleado.SelectedIndex = _empleados.Tipo_Employado_ID - 2;//xq los 2
primeros, están fuera de la búsqueda
txtUsuario.Text = _empleados.Usuario;
txtClave.Text = _empleados.Clave;
cmbEstados.SelectedIndex = _empleados.Estado_Employado_ID;

//Le voy a cambiar el color al texto de los textbox tmb, sino se ve todo en
DimGray
foreach (Control cnt in this.gbEmpleado.Controls)
{
    if (cnt is TextBox box)
    {
        TextBox t;
        t = box;
        if (t.Text != "Usuario" && t.Text != "Clave")
        {
            t.ForeColor = Color.Black;
        }
    }
}
if (cmbTipoEmpleado.SelectedIndex != 0)
{
    txtUsuario.Enabled = false;
    txtClave.Enabled = false;
}
edicionEmpleado = true;
}
private void GetEmpleados()
{
    //Primero limpio el grid. Para que no haya errores.
    //Sería como si hiciera un refresh.
    //Saco lo que haya y luego vuelvo a traer los datos
    dtgvEmpleados.Rows.Clear();

    dsTablaEmpleados = _bussinessEmpleados.GetEmpleados(buscar);
    buscar = string.Empty;

    if (dsTablaEmpleados.Tables[0].Rows.Count > 0)
    {
        foreach (DataRow dr in dsTablaEmpleados.Tables[0].Rows)
        {
            dtgvEmpleados.Rows.Add(dr[0].ToString(), dr[1], dr[2], dr[3], dr[4],
dr[5]);
        }
    }
}
private void ABMEmployee()
{
    switch (motivoEdicion)
    {
        case 0:
            //Edición de todos los datos del empleado
            //En caso de que no sea un alta, sino modificación,
            //se va a realizar desde acá
            EditarPersona();
    }
}

```

```

        EditarEmpleado();
        persona_Jor = Convert.ToInt32(dtgvEmpleados.CurrentRow.Cells[0].Value);

        break;
    case 1:
        //Edición solo de la jornada
        persona_Jor = Convert.ToInt32(dtgvEmpleados.CurrentRow.Cells[0].Value);

        break;
    case 2:
        //Blanqueo solo de usuario y clave.

        break;
    default:
        //Alta empleado
        AltaPersona();
        AltaEmpleado();
        _bussinessEmpleados.GetLastEmpleadoID(_empleados);
        persona_Jor = _empleados.Empleado_ID;
        break;
    }
    dsTablaEmpleados.Clear();
    GetEmpleados();
    ResetControls();
}

private void EditarPersona()
{
    //Es la edición de los datos personales
    //junto con los datos de empleado, excepto las claves
    int Persona_ID = _personas.Persona_ID;
    string Nombre = Convert.ToString(txtNombreEmpleado.Text);
    string Apellido = Convert.ToString(txtApellidoEmpleado.Text);
    int Tipo_Documento_ID = Convert.ToInt32(cmbTipoDocumentoEmpleado.SelectedValue);
    string Nro_documento = Convert.ToString(txtDocumentoEmpleado.Text);
    int Tipo_Sexo_ID = Convert.ToInt32(cmbSexoEmpleado.SelectedValue);
    string Nro_Teléfono = Convert.ToString(txtTeléfonoEmpleado.Text);
    string Nro_Alternativo, Mail, Observaciones;

    //Envío todo el objeto creado a la cada de negocios para ser enviado
    //a la bdd.

    //Acá verifico si el usuario ingresó datos nuevos o no a los campos
    //que reciben nulos.
    //Si no agregaron datos, los mandamos como nulos
    //caso contrario, se envía lo que haya en su correspondiente Textbox
    if (txtAlternativoEmpleado.Text != "Alternativo")
    {
        Nro_Alternativo = Convert.ToString(txtAlternativoEmpleado.Text);
    }
    else
    {
        Nro_Alternativo = string.Empty;
    }
    if (txtMailEmpleado.Text != "Mail")
    {
        Mail = Convert.ToString(txtMailEmpleado.Text);
    }
    else
    {
}

```

```

        Mail = string.Empty;
    }
    if (txtObservacionesEmpleado.Text != "Observaciones y/o consideraciones")
    {
        Observaciones = Convert.ToString(txtObservacionesEmpleado.Text);
    }
    else
    {
        Observaciones = string.Empty;
    }
    _metodosGenerales.EditarPersona(Persona_ID, Nombre,
                                    Apellido,
                                    Tipo_Documento_ID,
                                    Nro_documento,
                                    Tipo_Sexo_ID,
                                    Nro_Telefono,
                                    Nro_Alternativo,
                                    Mail,
                                    Observaciones);
}

private void AltaPersona()
{
    string Nombre = Convert.ToString(txtNombreEmpleado.Text);
    string Apellido = Convert.ToString(txtApellidoEmpleado.Text);
    int Tipo_Documento_ID = Convert.ToInt32(cmbTipoDocumentoEmpleado.SelectedValue);
    string Nro_documento = Convert.ToString(txtDocumentoEmpleado.Text);
    int Tipo_Sexo_ID = Convert.ToInt32(cmbSexoEmpleado.SelectedValue);
    string Nro_Telefono = Convert.ToString(txtTelefonoEmpleado.Text);
    DateTime FechaAlta = DateTime.Now;
    string Nro_Alternativo, Mail, Observaciones;

    //Envío todo el objeto creado a la cada de negocios para ser enviado
    //a la bdd.
    if (txtAlternativoEmpleado.Text != "Alternativo")
    {
        Nro_Alternativo = Convert.ToString(txtAlternativoEmpleado.Text);
    }
    else
    {
        Nro_Alternativo = string.Empty;
    }

    if (txtMailEmpleado.Text != "Mail")
    {
        Mail = Convert.ToString(txtMailEmpleado.Text);
    }
    else
    {
        Mail = string.Empty;
    }

    if (txtObservacionesEmpleado.Text != "Observaciones y/o consideraciones")
    {
        Observaciones = Convert.ToString(txtObservacionesEmpleado.Text);
    }
    else
    {
        Observaciones = string.Empty;
    }
    _metodosGenerales.AltaPersona(Nombre,

```

```

        Apellido,
        Tipo_Documento_ID,
        Nro_documento,
        Tipo_Sexo_ID,
        Nro_Teléfono,
        Nro_Alternativo,
        Mail,
        Observaciones,
        FechaAlta);
    }

private void EditarEmpleado()
{
    //Vamos a verificar si se ha modificado el combobox del tipo de empleado
    //cosa de saber si hace falta o no que tenga una clave.
    _empleados.Empleado_ID = _metodosGenerales.empleadoID;
    _empleados.Persona_ID = _metodosGenerales.personaID;
    _empleados.Estado_Empleado_ID = Convert.ToInt32(cmbEstados.SelectedValue);
    if (cmbTipoEmpleado.SelectedIndex == 0)
    {
        _empleados.Tipo_Empleado_ID = Convert.ToInt32(cmbTipoEmpleado.SelectedValue);
        _empleados.Usuario = txtUsuario.Text.ToString();
        _empleados.Clave = clave;
    }
    else
    {
        //Caso contrario, si el perfil no requiere clave, se envían los campos vacíos.
        _empleados.Usuario = string.Empty;
        _empleados.Clave = string.Empty;
    }

    //Mando todo a la capa de negocio
    _bussinessEmpleados.EditarEmpleado(_empleados);
}

private void AltaEmpleado()
{
    _empleados.Persona_ID = _metodosGenerales.persona_ID;
    _empleados.Tipo_Empleado_ID = Convert.ToInt32(cmbTipoEmpleado.SelectedValue);
    _empleados.Estado_Empleado_ID = Convert.ToInt32(cmbEstados.SelectedValue);
    if (cmbTipoEmpleado.SelectedIndex == 0)
    {
        _empleados.Tipo_Empleado_ID = Convert.ToInt32(cmbTipoEmpleado.SelectedValue);
        _empleados.Usuario = txtUsuario.Text.ToString();
        _empleados.Clave = clave;
    }
    else
    {
        //Caso contrario, si el perfil no requiere clave, se envían los campos vacíos.
        _empleados.Usuario = string.Empty;
        _empleados.Clave = string.Empty;
    }
    //Envío toda la data a la capa de negocio para ser mandada a la bdd.
    _bussinessEmpleados.AltaEmpleado(_empleados);
}

private void EncriptarClaveClave()
{
    //Encriptamos la clave con un HASH256
    //Y la asignamos al campo de la entidad correspondiente
    clave = EncriptamientoSHA256.GetSHA256(txtClave.Text.ToString());
}

```

```

private void VerificarClave()
{
    if (cmbTipoEmpleado.SelectedIndex == 0)
    {
        if (txtUsuario.Text != "Usuario" && txtClave.Text != "Clave")
        {
            //Si está todo OK.
            //Vamos a encriptar la clave y consultar si es una edición o alta.
            EncriptarClaveClave();
            //Tambien asigno la clave sin encriptación a otra variable.
            string claveSinCypt = txtClave.Text;

            if (motivoEdicion == 0)
            {
                //Si es una edición, hay que comparar si la clave nueva es != a la de
la bdd
                if (claveSinCypt == _empleados.Clave)
                {
                    _empleados.Clave = claveSinCypt;
                }
                else
                {
                    _empleados.Clave = clave;
                }
            }
            else
            {
                _empleados.Clave = clave;
            }
            claveOK = true;
        }
        else
        {
            MessageBox.Show("El usuario y/o la clave están mal registrados. Por favor,
asigne usuario y clave validos.", "Usuario y/o clave mal registrados");
            claveOK = false;
        }
    }
}

private void Tipos_Documentos()
{
    //llamo al método para traer los tipos de documentos y los asigno al combobox
    _metodosGenerales.Bring_Tipos_Documentos();
    cmbTipoDocumentoEmpleado.DataSource = _metodosGenerales.DtTipos_Documentos;
    cmbTipoDocumentoEmpleado.DisplayMember = "Tipo";
    cmbTipoDocumentoEmpleado.ValueMember = "Tipo_documento_ID";
}

private void Tipos_Sexos()
{
    //llamo al método para traer los tipos de sexos y los asigno al combobox
    _metodosGenerales.Bring_Tipos_Sexos();
    cmbSexoEmpleado.DataSource = _metodosGenerales.DtTipos_Sexos;
    cmbSexoEmpleado.DisplayMember = "Sexo"; //Pero solo esta columna es la que muestro
    cmbSexoEmpleado.ValueMember = "Tipo_Sexo_ID";
}

private void Tipos_Empleados()
{
    //llamo al método para traer los tipos de Empleados y los asigno al combobox
    _metodosGenerales.Bring_Tipos_Employados();
    cmbTipoEmpleado.DataSource = _metodosGenerales.DtTipos_Employados;
    cmbTipoEmpleado.DisplayMember = "Tipo"; //Pero solo esta columna es la que muestro
}

```

```

cmbTipoEmpleado.ValueMember = "Tipo_Emppleado_ID";
}
private void Estados_Employees()
{
    //llamo al método para traer todos los estados disponibles en la base de datos.
    _metodosGenerales.GetEstadosEmployees();
    cmbEstados.DataSource = _metodosGenerales.DtEstados_Employees;
    cmbEstados.Tag = "Estado inicial del empleado";
    cmbEstados.DisplayMember = "Estado_Employee"; //Pero solo esta columna es la que
muestro
    cmbEstados.ValueMember = "Estado_Employee_ID";
}
private void ValidarCamposVacios()
{
    if (string.IsNullOrEmpty(txtNombreEmployee.Text) ||
        string.IsNullOrEmpty(txtApellidoEmployee.Text) ||
        string.IsNullOrEmpty(txtDocumentoEmployee.Text) ||
        string.IsNullOrEmpty(txtTelefonoEmployee.Text))
    {
        camposObligatoriosVacios = true;
    }
    else
    {
        camposObligatoriosVacios = false;
    }
}
private void ValidarContenido()
{
    foreach (Control txt in this.gbEmployee.Controls)
    {
        if (txt is TextBox)
        {
            if (txt.Text == "Nombre" ||
                txt.Text == "Apellido" ||
                txt.Text == "Documento" ||
                txt.Text == "Teléfono")
            {
                MessageBox.Show("Existen campos sin completar. Por favor, rellene
todos los campos obligatorios", "Campos obligatorios sin llenar.", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                contenidoErroneo = true;
                break;
            }
            else
            {
                contenidoErroneo = false;
            }
        }
    }
}
private void ResetControls()
{
    //luego el groupbox general, lo volvemos al estado inicial.
    txtNombreEmployee.Text = "Nombre";
    txtApellidoEmployee.Text = "Apellido";
    txtDocumentoEmployee.Text = "Documento";
    txtTelefonoEmployee.Text = "Teléfono";
    txtAlternativoEmployee.Text = "Alternativo";
    txtMailEmployee.Text = "Mail";
    txtObservacionesEmployee.Text = "Observaciones y/o consideraciones";
    txtUsuario.Text = "Usuario";
    txtClave.Text = "Clave";
}

```

```
cmbTipoDocumentoEmpleado.SelectedItem = 0;
cmbSexoEmpleado.SelectedItem = 0;
cmbTipoEmpleado.SelectedItem = 0;
foreach (Control txt in this.gbEmpleado.Controls)
{
    if (txt is TextBox box)
    {
        txt.ForeColor = Color.DimGray;
    }
}
}

private void HabilitarClave()
{
    if (cmbTipoEmpleado.SelectedIndex == 0)
    {
        txtUsuario.Enabled = true;
        txtClave.Enabled = true;
    }
    else
    {
        txtUsuario.Enabled = false;
        txtClave.Enabled = false;
    }
}

public void FormIncompleto(bool formIncompleto)
{
    foreach (Control cntl in this.gbEmpleado.Controls)
    {
        if (cntl is TextBox box)
        {
            TextBox t;
            t = box;
            if (t.Text == string.Empty || t.ForeColor == Color.DimGray)
            {
                DialogResult result = MessageBox.Show("Hay campos sin completar.  
¿Quiere salir de todas formas?", "Campos incompletos", MessageBoxButtons.YesNo,
                MessageBoxIcon.Warning);
                if (result == DialogResult.Yes)
                {
                    formIncompleto = false;
                }
                else
                {
                    formIncompleto = true;
                }
                break;
            }
            else
            {
                formIncompleto = false;
            }
        }
    }
}

private void EditarDatos()
{
    BuscarEmpleado();
    HabilitarClave();
```

```
}

private void EditarLogin()
{
    _empleados.Usuario = txtUsuario.Text.ToString();
    EncriptarClaveClave();
    _bussinessEmpleados.EditarClave(_empleados);
}

private void VerificarCoincidencias()
{
    if (motivoEdicion != 0)
    {
        int id = _empleados.Persona_ID;
        string documento = txtDocumentoEmpleado.Text.ToString();
        _bussinessPersonas.BuscarCoincidencias(id, documento, _personas);
        if (_personas.Persona_ID == id && _personas.Nro_documento == documento)
        {
            personaRegistrada = true;
        }
        else
        {
            personaRegistrada = false;
        }
    }
}

#endregion

#region Eventos de Foco en TextBox

private void txtNombreEmpleado_Enter(object sender, System.EventArgs e)
{
    //Primero preguntamos si el texto dice Nombre
    //Al principio siempre lo va a decir, porque así
    //inicializa el form. Por lo que lo vamos a vaciar
    //y cambiar el color de la letra.
    //En caso de que no diga literalmente Nombre
    //Se va a dejar como está, ya que puede ser
    //un texto que el usuario escribió

    if (txtNombreEmpleado.Text == "Nombre")
    {
        txtNombreEmpleado.Text = string.Empty;
        txtNombreEmpleado.ForeColor = Color.Black;
    }
}

private void txtNombreEmpleado_Leave(object sender, System.EventArgs e)
{
    //Por el lado contrario, cuando se quita el foco
    //del control, lo volvemos a su estado y color inicial

    if (string.IsNullOrEmpty(txtNombreEmpleado.Text))
    {
        txtNombreEmpleado.Text = "Nombre";
        txtNombreEmpleado.ForeColor = Color.DimGray;
    }
}

/*Así igual para los siguientes eventos dentro de esta región*/
private void txtApellidoEmpleado_Enter(object sender, System.EventArgs e)
```

```
{  
    if (txtApellidoEmpleado.Text == "Apellido")  
    {  
        txtApellidoEmpleado.Text = string.Empty;  
        txtApellidoEmpleado.ForeColor = Color.Black;  
    }  
}  
  
private void txtApellidoEmpleado_Leave(object sender, System.EventArgs e)  
{  
    if (string.IsNullOrEmpty(txtApellidoEmpleado.Text))  
    {  
        txtApellidoEmpleado.Text = "Apellido";  
        txtApellidoEmpleado.ForeColor = Color.DimGray;  
    }  
}  
private void txtDocumentoEmpleado_Enter(object sender, System.EventArgs e)  
{  
    if (txtDocumentoEmpleado.Text == "Documento")  
    {  
        txtDocumentoEmpleado.Text = string.Empty;  
        txtDocumentoEmpleado.ForeColor = Color.Black;  
    }  
}  
private void txtDocumentoEmpleado_Leave(object sender, System.EventArgs e)  
{  
    if (string.IsNullOrEmpty(txtDocumentoEmpleado.Text))  
    {  
        txtDocumentoEmpleado.Text = "Documento";  
        txtDocumentoEmpleado.ForeColor = Color.DimGray;  
    }  
    else  
    {  
        VerificarCoincidencias();  
    }  
}  
private void txtTelefonoEmpleado_Enter(object sender, System.EventArgs e)  
{  
    if (txtTelefonoEmpleado.Text == "Teléfono")  
    {  
        txtTelefonoEmpleado.Text = string.Empty;  
        txtTelefonoEmpleado.ForeColor = Color.Black;  
    }  
}  
private void txtTelefonoEmpleado_Leave(object sender, System.EventArgs e)  
{  
    if (string.IsNullOrEmpty(txtTelefonoEmpleado.Text))  
    {  
        txtTelefonoEmpleado.Text = "Teléfono";  
        txtTelefonoEmpleado.ForeColor = Color.DimGray;  
    }  
}  
  
private void txtAlternativoEmpleado_Enter(object sender, System.EventArgs e)  
{  
    if (txtAlternativoEmpleado.Text == "Alternativo")  
    {  
        txtAlternativoEmpleado.Text = string.Empty;  
        txtAlternativoEmpleado.ForeColor = Color.Black;  
    }  
}
```

```
private void txtAlternativoEmpleado_Leave(object sender, System.EventArgs e)
{
    if (string.IsNullOrEmpty(txtAlternativoEmpleado.Text))
    {
        txtAlternativoEmpleado.Text = "Alternativo";
        txtAlternativoEmpleado.ForeColor = Color.DimGray;
    }
}

private void txtMailEmpleado_Enter(object sender, System.EventArgs e)
{
    if (txtMailEmpleado.Text == "Mail")
    {
        txtMailEmpleado.Text = string.Empty;
        txtMailEmpleado.ForeColor = Color.Black;
    }
}

private void txtMailEmpleado_Leave(object sender, System.EventArgs e)
{
    if (string.IsNullOrEmpty(txtMailEmpleado.Text))
    {
        txtMailEmpleado.Text = "Mail";
        txtMailEmpleado.ForeColor = Color.DimGray;
    }
}

private void txtObservacionesEmpleado_Enter(object sender, System.EventArgs e)
{
    if (txtObservacionesEmpleado.Text == "Observaciones y/o consideraciones")
    {
        txtObservacionesEmpleado.Text = string.Empty;
        txtObservacionesEmpleado.ForeColor = Color.Black;
    }
}

private void txtObservacionesEmpleado_Leave(object sender, System.EventArgs e)
{
    if (string.IsNullOrEmpty(txtObservacionesEmpleado.Text))
    {
        txtObservacionesEmpleado.Text = "Observaciones y/o consideraciones";
        txtObservacionesEmpleado.ForeColor = Color.DimGray;
    }
}

private void txtBuscarEmpleado_Enter(object sender, EventArgs e)
{
    if (txtBuscarEmpleado.Text == "Buscar")
    {
        txtBuscarEmpleado.Text = string.Empty;
        txtBuscarEmpleado.ForeColor = Color.Black;
    }
}

private void txtBuscarEmpleado_Leave(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(txtBuscarEmpleado.Text))
    {
        txtBuscarEmpleado.Text = "Buscar";
        txtBuscarEmpleado.ForeColor = Color.DimGray;
    }
}
```

```
        }

    private void txtUsuario_Enter(object sender, EventArgs e)
    {
        if (txtUsuario.Text == "Usuario")
        {
            txtUsuario.Text = string.Empty;
            txtUsuario.ForeColor = Color.Black;
        }
    }

    private void txtUsuario_Leave(object sender, EventArgs e)
    {
        if (string.IsNullOrEmpty(txtUsuario.Text))
        {
            txtUsuario.Text = "Usuario";
            txtUsuario.ForeColor = Color.DimGray;
        }
    }

    private void txtClave_Enter(object sender, EventArgs e)
    {
        if (txtClave.Text == "Clave")
        {
            txtClave.Text = string.Empty;
            txtClave.ForeColor = Color.Black;
        }
    }

    private void txtClave_Leave(object sender, EventArgs e)
    {
        if (string.IsNullOrEmpty(txtClave.Text))
        {
            txtClave.Text = "Clave";
            txtClave.ForeColor = Color.DimGray;
        }
    }

#endregion

#region Eventos KeyPress en Textbox

    private void txtNumDocumentoEmpleado_KeyPress(object sender, KeyPressEventArgs e)
    {
        string strTexto = txtDocumentoEmpleado.Text;
        _restricciones.SoloNumeros(e, strTexto);
    }

    private void txtTelefonoEmpleado_KeyPress(object sender, KeyPressEventArgs e)
    {
        string strTexto = txtTelefonoEmpleado.Text;
        _restricciones.SoloNumeros(e, strTexto);
    }

    private void txtAlternativoEmpleado_KeyPress(object sender, KeyPressEventArgs e)
    {
        string strTexto = txtAlternativoEmpleado.Text;
        _restricciones.SoloNumeros(e, strTexto);
    }

    private void txtBuscarEmpleado_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (e.KeyChar == Convert.ToChar(Keys.Enter))
        {
            buscar = txtBuscarEmpleado.Text;
            GetEmpleados();
        }
    }


```

```

        }

    }

    private void cmbTipoEmpleado_SelectedIndexChanged(object sender, EventArgs e)
    {
        HabilitarClave();
    }

#endregion

#region Alta Empleado
private void btnAltaEmpleado_Click(object sender, EventArgs e)
{
    //Valido los campos obligatorios
    ValidarCamposVacios();
    if (!camposObligatoriosVacios)
    {
        //Valido si el contenido de los campos es válido
        ValidarContenido();
        if (!contenidoErroneo)
        {
            //Verifico si hay clientes con los datos registrados
            VerificarCoincidencias();
            if (!personaRegistrada)
            {
                //Veo si el empleado es el que tiene acceso al programa
                if (Convert.ToInt32(cmbTipoEmpleado.SelectedValue) == 2)
                {
                    VerificarClave();
                    if (claveOK)
                    {
                        //Hacemos la edición de la persona.
                        ABMEmppleado();

                        //Luego verificamos si va con jornada.
                        if (chkJornadaEmpleados.Checked)
                        {
                            CargarJornada();
                        }
                        motivoEdicion = 3;
                    }
                }
            }
            else
            {
                ABMEmppleado();

                //Luego verificamos si va con jornada.
                if (chkJornadaEmpleados.Checked)
                {
                    CargarJornada();
                    motivoEdicion = 3;
                }
            }
        }
        else
        {
            //Vamos a mostrar un mensaje, en caso de que esta persona ya esté
            registrada.
            MessageBox.Show("Existe un registro idéntico del dni de esta persona.
Solo pueden hacerse modificaciones", "Coincidencia encontrada", MessageBoxButtons.OK,
MessageBoxIcon.Information);
        }
    }
}

```

```
        }
    }
}

MessageBox.Show("Campos obligatorios sin completar. Por favor, complete todos
los campos requeridos", "Campos vacíos", MessageBoxButtons.OK);
}
}

#endregion

#region Edición de datos de empleado.

private void editarEmpleado_Click(object sender, EventArgs e)
{
    motivoEdicion = 0;
    if (edicionEmpleado)
    {
        DialogResult result = MessageBox.Show("Tiene una edición en proceso. " +
            "¿Desea cancelarla y editar otro empleado?", "Edición en proceso",
            MessageBoxButtons.OKCancel);
        if (result == DialogResult.OK)
        {
            EditarDatos();
        }
    }
    else
    {
        EditarDatos();
    }
}
private void editarClave_Click(object sender, EventArgs e)
{
    motivoEdicion = 2;
    foreach (Control ctrl in this.Controls)
    {
        if (ctrl is TextBox box)
        {
            TextBox t;
            t = box;
            if (t != txtUsuario && t != txtClave)
            {
                t.Enabled = false;
            }
        }
    }
    EditarDatos();
    EditarLogin();
}
private void editarJornada_Click(object sender, EventArgs e)
{
    motivoEdicion = 1;
    CargarJornada();
}

#endregion
}
```

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

namespace Gym
{
    class EncriptamientoSHA256
    {
        public static string GetSHA256(string clave)
        {
            //Este método devuelve un hash de 64bytes
            //de la clave que le enviamos como argumento desde cualquier otra clase
            //en la que se invoque este método.
            SHA256 sha256 = SHA256Managed.Create();
            ASCIIEncoding encoding = new ASCIIEncoding();
            byte[] stream = null;
            StringBuilder sb = new StringBuilder();
            stream = sha256.ComputeHash(encoding.GetBytes(clave));
            for (int i = 0; i < stream.Length; i++) sb.AppendFormat("{0:x2}", stream[i]);
            return sb.ToString();
        }
    }
}
```

```
using BussinessLayer;
using Entities;
using System;
using System.Data;
using System.Drawing;
using System.Runtime.InteropServices;
using System.Windows.Forms;

namespace Gym
{
    public partial class Jornadas : Form
    {
        #region Instancias
        //Entidades
        private Jornadas_Empleados _jornadas_Empleados;
        private Entities.Jornadas_Planes _jornadasPlanes;
        private Entities.Employees _empleados;

        //Capa de Negocio
        private readonly BussinessJornadas _bussinessJornadas;
        private readonly BussinessEmployees _bussinessEmployees;

        //Clases internas
        private readonly MetodosGenerales _metodosGenerales;
        private readonly Restricciones _restricciones;

        #endregion

        #region Variables
        //variables de tipo table.
        public DataTable DtJornadas = new DataTable();

        //otras.
        private string desde;
        private string hasta;
        private bool todosChk = false;
        private int check;
        private int contador = 0;
        private string formatoHora = "El formato de la hora es este: hh:mm";
        private int pTodosId = -1;
        private int pLunesId = -1;
        private int pMartesId = -1;
        private int pMiercolesId = -1;
        private int pJuevesId = -1;
        private int pViernesId = -1;
        private int pSabadoId = -1;
        private bool contenidoIncorrecto = false;
        private int empleado_ID;
        private bool cargarJornada;
        private int idJornada;
        private int contadorJornadas;
        private bool esEmpleado;
        private bool darleLaBaja;
        private string hora;

        #endregion

        #region Load
        public Jornadas(int jornadaID, bool empleado, bool darBaja)
        {
            InitializeComponent();
        }
    }
}
```

```

_bussinessJornadas = new BussinessJornadas();
_jornadas_Empieados = new Jornadas_Empieados();
_metodosGenerales = new MetodosGenerales();
_empleados = new Entities.Empieados();
_bussinessEmpleados = new BussinessEmpleados();
_restricciones = new Restricciones();
_jornadasPlanes = new Entities.Jornadas_Planes();

idJornada = jornadaID;
esEmpleado = empleado;
darleLaBaja = darBaja;

DarDeBajaLaJornada();
}

#endregion

#region Métodos encapsulados
private void DarDeBajaLaJornada()
{
    if (darleLaBaja)
    {
        if (esEmpleado)
        {
            _bussinessJornadas.EliminarJornadaEmpleado(idJornada);
        }
        else
        {
            _bussinessJornadas.EliminarJornadaPlan(idJornada);
        }
        MessageBox.Show("Las jornadas asociadas, se han eliminado también",
"Eliminación de jornadas");
        this.Close();
    }
    else
    {
        if (esEmpleado)
        {
            GetJornadaEmpleado();
        }
        else
        {
            EsJornadaDePlanes();
            GetJornadaPlan();
        }
    }
}
private void EsJornadaDePlanes()
{
    foreach (Control ctrl in gbJornadaEmpleado.Controls)
    {
        if (ctrl is TextBox box)
        {
            TextBox t;
            t = box;
            t.ReadOnly = true;
        }
        if (ctrl is CheckBox chk)
        {
            CheckBox c;
            c = chk;
        }
    }
}

```

```

        c.Enabled = false;
    }
    if (ctrl is Button btn)
    {
        Button b;
        b = btn;
        b.Enabled = false;
    }
}
btnAltaJornada.Enabled = false;
lblCancelar.Text = "Cerrar";
ActiveControl = lblCancelar;
}
private void GetJornadaPlan()
{
    _jornadasPlanes.Plan_ID = idJornada;
    //Una vez asignado, busco la jornada
    //Y la asigno a un datatable, para poder manejar el contenido.
    DtJornadas = _bussinessJornadas.GetJornadaPlan(_jornadasPlanes);

    if (DtJornadas.Rows.Count > 0)
    {
        cargarJornada = true;
        //cargo la jornada en los textbox correspondientes
        CargarJornada();
    }
    else
    {
        cargarJornada = false;
    }
}
private void GetJornadaEmpleado()
{
    _jornadas_Employados.Empleado_ID = idJornada;
    //Una vez asignado, busco la jornada
    //Y la asigno a un datatable, para poder manejar el contenido.
    DtJornadas = _bussinessJornadas.GetJornadaEmpleado(_jornadas_Employados);

    if (DtJornadas.Rows.Count > 0)
    {
        cargarJornada = true;
        //cargo la jornada en los textbox correspondientes
        CargarJornada();
    }
    else
    {
        cargarJornada = false;
    }
}

private void CargarJornada()
{
    foreach (DataRow dr in DtJornadas.Rows)
    {
        //Vamos a switchear cada vuelta para que vaya
        //colocando los datos cada vuelta donde corresponda.
        switch (dr[2].ToString())
        {
            case "Todos":
                chkTodos.Checked = true;

```

```

//Este es el Id de la jornada que se carga
pTodosId = Convert.ToInt32(dr[0]);
txtDesdeLunes.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[3].ToString()));
txtHastaLunes.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[4].ToString()));
txtDesdeLunes.ForeColor = Color.Black;
txtHastaLunes.ForeColor = Color.Black;
VerificarChk();
break;
case "Lunes":
    txtDesdeLunes.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[3].ToString()));
Convert.ToString(dr[3]);
    txtHastaLunes.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[4].ToString()));
    txtDesdeLunes.ForeColor = Color.Black;
    txtHastaLunes.ForeColor = Color.Black;
    pLunesId = Convert.ToInt32(dr[0]);
    chkLunes.Checked = true;
    break;
case "Martes":
    txtDesdeMartes.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[3].ToString()));
    txtHastaMartes.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[4].ToString()));
    txtDesdeMartes.ForeColor = Color.Black;
    txtHastaMartes.ForeColor = Color.Black;
    pMartesId = Convert.ToInt32(dr[0]);
    chkMartes.Checked = true;
    break;
case "Miércoles":
    txtDesdeMiércoles.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[3].ToString()));
    txtHastaMiércoles.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[4].ToString()));
    txtDesdeMiércoles.ForeColor = Color.Black;
    txtHastaMiércoles.ForeColor = Color.Black;
    pMiércolesId = Convert.ToInt32(dr[0]);
    chkMiércoles.Checked = true;
    break;
case "Jueves":
    txtDesdeJueves.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[3].ToString()));
    txtHastaJueves.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[4].ToString()));
    txtDesdeJueves.ForeColor = Color.Black;
    txtHastaJueves.ForeColor = Color.Black;
    pJuevesId = Convert.ToInt32(dr[0]);
    chkJueves.Checked = true;
    break;
case "Viernes":
    txtDesdeViernes.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[3].ToString()));
    txtHastaViernes.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[4].ToString()));
    txtDesdeViernes.ForeColor = Color.Black;
    txtHastaViernes.ForeColor = Color.Black;
    pViernesId = Convert.ToInt32(dr[0]);
    chkViernes.Checked = true;
    break;
case "Sábado":
    txtDesdeSabado.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[3].ToString()));

```

```
        txtHastaSabado.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[4].ToString())));
        txtDesdeSabado.ForeColor = Color.Black;
        txtHastaSabado.ForeColor = Color.Black;
pSabadoId = Convert.ToInt32(dr[0]);
chkSabado.Checked = true;
break;
}
}
}
private void VerificarChk()
{
    switch (check)
    {
        case 0:
            ActivarTodoChk();
            break;
        case 1:
            ChkDias();
            break;
        case 2:
            ChkDias();
            break;
        case 3:
            ChkDias();
            break;
        case 4:
            ChkDias();
            break;
        case 5:
            ChkDias();
            break;
        case 6:
            ChkDias();
            break;
    }
    contador = 0;
}

private void ChkDias()
{
    RevisarTodosChk();
    if (chkTodos.Checked)
    {
        chkTodos.Checked = false;
        todosChk = false;
    }
    else
    {
        if (contador > 5 && !chkTodos.Checked)
        {
            ActivarTodoChk();
        }
    }
}
private void ActivarTodoChk()
{
    if (!chkTodos.Checked && contador < 7 && !todosChk)
    {
        foreach (Control chk in gbJornadaEmpleado.Controls)
        {
```

```

        if (chk is CheckBox box)
        {
            CheckBox c;
            c = box;
            c.Checked = true;
        }
    }
    todosChk = true;
}
else
{
    if (todosChk)
    {
        //Si el chk que dice "Seleccionar todos" está seleccionado
        //Entonces todos los otros chk se van a seleccionar
        //Caso contrario, se les quitará el check
        foreach (Control chk in gbJornadaEmpleado.Controls)
        {
            if (chk is CheckBox box)
            {
                CheckBox c;
                c = box;
                //Deshabilito el check en cada checkbox
                c.Checked = false;
            }
        }
        todosChk = false;
    }
    else
    {
        foreach (Control chk in gbJornadaEmpleado.Controls)
        {
            if (chk is CheckBox box)
            {
                CheckBox c;
                c = box;
                c.Checked = true;
            }
        }
        todosChk = true;
    }
}
private void RevisarTodosChk()
{
    foreach (Control chk in gbJornadaEmpleado.Controls)
    {
        if (chk is CheckBox box)
        {
            CheckBox c;
            c = box;
            if (c.Checked)
            {
                contador++;
            }
        }
    }
}

private void AsignacionHoras()
{

```

```

int contadorDesde = desde.Length;
int contadorHasta = hasta.Length;

if (contadorDesde <= 5 && contadorHasta <= 5)
{
    _jornadas_Empieados.Desde_Hora = _metodosGenerales.ConvertirHoraAfecha(desde);
    _jornadas_Empieados.Hasta_Hora = _metodosGenerales.ConvertirHoraAfecha(hasta);
}
else
{
    MessageBox.Show("El campo de horas, está mal registrado. Desde registrarse con este formato: ##:##", "Formato incorrecto", MessageBoxButtons.OK, MessageBoxIcon.Warning);
}
}

private void GuardarJornada()
{
    foreach (Control chk in this.gbJornadaEmpleado.Controls)
    {
        if (chk is CheckBox box)
        {
            CheckBox c;
            c = box;
            //Y luego de cada checkbox le asignamos el horario.
            //Lunes
            if (c.Checked)
            {
                switch (c.Text)
                {
                    case "Lunes":
                        _jornadas_Empieados.Dia = "Lunes";
                        _jornadas_Empieados.Jornada_Empieado_ID = pLunesId;
                        desde = txtDesdeLunes.Text.ToString();
                        hasta = txtHastaLunes.Text.ToString();
                        AsignacionHoras();
                        break;
                    case "Martes":
                        _jornadas_Empieados.Dia = "Martes";
                        _jornadas_Empieados.Jornada_Empieado_ID = pMartesId;
                        desde = txtDesdeMartes.Text.ToString();
                        hasta = txtHastaMartes.Text.ToString();
                        AsignacionHoras();
                        break;
                    case "Miercoles":
                        _jornadas_Empieados.Dia = "Miercoles";
                        _jornadas_Empieados.Jornada_Empieado_ID = pMiercolesId;
                        desde = txtDesdeMiercoles.Text.ToString();
                        hasta = txtHastaMiercoles.Text.ToString();
                        AsignacionHoras();
                        break;
                    case "Jueves":
                        _jornadas_Empieados.Dia = "Jueves";
                        _jornadas_Empieados.Jornada_Empieado_ID = pJuevesId;
                        desde = txtDesdeJueves.Text.ToString();
                        hasta = txtHastaJueves.Text.ToString();
                        AsignacionHoras();
                        break;
                    case "Viernes":
                        _jornadas_Empieados.Dia = "Viernes";
                        _jornadas_Empieados.Jornada_Empieado_ID = pViernesId;
                        desde = txtDesdeViernes.Text.ToString();
                }
            }
        }
    }
}

```

```

        hasta = txtHastaViernes.Text.ToString();
        AsignacionHoras();
        break;
    case "Sabado":
        _jornadas_Employados.Dia = "Sabado";
        _jornadas_Employados.Jornada_Employee_ID = pSabadoId;
        desde = txtDesdeSabado.Text.ToString();
        hasta = txtHastaSabado.Text.ToString();
        AsignacionHoras();
        break;
    }
    _jornadas_Employados.Esto = "A";
    _jornadas_Employados.Employee_ID = idJornada;
    if (_jornadas_Employados.Jornada_Employee_ID != -1)
    {
        _bussinessJornadas.EditarJornadaEmployee(_jornadas_Employados);
    }
    else
    {
        _bussinessJornadas.AltaJornadaEmployee(_jornadas_Employados);
    }
    desde = string.Empty;
    hasta = string.Empty;
}
}
}
}
}

private void JornadasManagement()
{
    //Acá editamos
    if (cargarJornada)
    {
        GuardarJornada();
    }
    else
    {
        //Acá creamos:

        //Ahora tenemos 2 opciones.
        // 1 es que se haya seleccionado el check "todos"
        if (todosChk)
        {
            _jornadas_Employados.Employee_ID = idJornada;
            _jornadas_Employados.Dia = "Todos";
            _jornadas_Employados.Esto = "A";
            desde = txtDesdeLunes.Text.ToString();
            hasta = txtHastaLunes.Text.ToString();
            AsignacionHoras();

            _bussinessJornadas.AltaJornadaEmployee(_jornadas_Employados);
        }
        else
        {
            //Y la otra es que se hayan seleccionado varios días
            //Vamos a verificar los checkbox que se hayan seleccionado
            GuardarJornada();

        }
    }
}
}

private void EliminarJornada(int jornadaID)

```

```

    {
        _bussinessJornadas.EliminarJornadaEmpleado(jornadaID);
    }

    private void RevisarContenido()
    {
        if (chkTodos.Checked)
        {
            if (txtDesdeLunes.Text != "hh:mm" ||
!string.IsNullOrEmpty(txtDesdeLunes.Text))
            {
                contenidoIncorrecto = false;
            }
            else
            {
                contenidoIncorrecto = true;
            }
        }
        else
        {
            foreach (Control ctrl in gbJornadaEmpleado.Controls)
            {
                if (ctrl is TextBox txt)
                {
                    //Si el contenido de txt es igual a "hh:mm" o es nulo o está vacío
                    if (txt.Text == "hh:mm" || string.IsNullOrEmpty(txt.Text))
                    {
                        //entonces voy a ver si el checkbox que le corresponde está
                        seleccionado
                        foreach (Control ctl in gbJornadaEmpleado.Controls)
                        {
                            if (ctl is CheckBox chk)
                            {
                                //si está seleccionado y es el que corresponde..
                                if (chk.Checked && txt.Name.Contains(chk.Text))
                                {
                                    //entonces el contenido está mal registrado y hay que
                                    registrar bien
                                    //el contenido de la hora en cada cuadro.
                                    contenidoIncorrecto = true;
                                    break;
                                }
                            }
                        }
                    }
                    if (contenidoIncorrecto)
                    {
                        break;
                    }
                }
                else
                {
                    contenidoIncorrecto = false;
                }
            }
        }
    }

    private void VerificarFormatoHora()
    {
        foreach (Control ctrl in gbJornadaEmpleado.Controls)
        {
    
```

```
        if (ctrl is TextBox box)
        {
            TextBox t;
            t = box;
            if (t.Text.Length < 5)
            {
                contenidoIncorrecto = true; ;
                break;
            }
            else
            {
                contenidoIncorrecto = false;
            }
        }
    }

#endregion

#region Focus TextBox

//Lunes
private void txtDesdeLunes_Enter(object sender, EventArgs e)
{
    if (txtDesdeLunes.Text == "hh:mm")
    {
        txtDesdeLunes.Text = string.Empty;
        txtDesdeLunes.ForeColor = Color.Black;
    }
}
private void txtDesdeLunes_Leave(object sender, EventArgs e)
{
    //Por el lado contrario, cuando se quita el foco
    //del control, lo volvemos a su estado y color inicial

    if (string.IsNullOrEmpty(txtDesdeLunes.Text))
    {
        txtDesdeLunes.Text = "hh:mm";
        txtDesdeLunes.ForeColor = Color.DimGray;
    }
}
private void txtHastaLunes_Enter(object sender, EventArgs e)
{
    if (txtHastaLunes.Text == "hh:mm")
    {
        txtHastaLunes.Text = string.Empty;
        txtHastaLunes.ForeColor = Color.Black;
    }
}
private void txtHastaLunes_Leave(object sender, EventArgs e)
{
    //Por el lado contrario, cuando se quita el foco
    //del control, lo volvemos a su estado y color inicial

    if (string.IsNullOrEmpty(txtHastaLunes.Text))
    {
        txtHastaLunes.Text = "hh:mm";
        txtHastaLunes.ForeColor = Color.DimGray;
    }
}
```

```
//Martes
private void txtDesdeMartes_Enter(object sender, EventArgs e)
{
    if (txtDesdeMartes.Text == "hh:mm")
    {
        txtDesdeMartes.Text = string.Empty;
        txtDesdeMartes.ForeColor = Color.Black;
    }
}
private void txtDesdeMartes_Leave(object sender, EventArgs e)
{
    //Por el lado contrario, cuando se quita el foco
    //del control, lo volvemos a su estado y color inicial

    if (string.IsNullOrEmpty(txtDesdeMartes.Text))
    {
        txtDesdeMartes.Text = "hh:mm";
        txtDesdeMartes.ForeColor = Color.DimGray;
    }
}
private void txtHastaMartes_Enter(object sender, EventArgs e)
{
    if (txtHastaMartes.Text == "hh:mm")
    {
        txtHastaMartes.Text = string.Empty;
        txtHastaMartes.ForeColor = Color.Black;
    }
}
private void txtHastaMartes_Leave(object sender, EventArgs e)
{
    //Por el lado contrario, cuando se quita el foco
    //del control, lo volvemos a su estado y color inicial

    if (string.IsNullOrEmpty(txtHastaMartes.Text))
    {
        txtHastaMartes.Text = "hh:mm";
        txtHastaMartes.ForeColor = Color.DimGray;
    }
}

//Miércoles
private void txtDesdeMiercoles_Enter(object sender, EventArgs e)
{
    if (txtDesdeMiercoles.Text == "hh:mm")
    {
        txtDesdeMiercoles.Text = string.Empty;
        txtDesdeMiercoles.ForeColor = Color.Black;
    }
}
private void txtDesdeMiercoles_Leave(object sender, EventArgs e)
{
    //Por el lado contrario, cuando se quita el foco
    //del control, lo volvemos a su estado y color inicial

    if (string.IsNullOrEmpty(txtDesdeMiercoles.Text))
    {
        txtDesdeMiercoles.Text = "hh:mm";
        txtDesdeMiercoles.ForeColor = Color.DimGray;
    }
}
private void txtHastaMiercoles_Enter(object sender, EventArgs e)
```

```
{  
    if (txtHastaMiercoles.Text == "hh:mm")  
    {  
        txtHastaMiercoles.Text = string.Empty;  
        txtHastaMiercoles.ForeColor = Color.Black;  
    }  
}  
}  
private void txtHastaMiercoles_Leave(object sender, EventArgs e)  
{  
    //Por el lado contrario, cuando se quita el foco  
    //del control, lo volvemos a su estado y color inicial  
  
    if (string.IsNullOrEmpty(txtHastaMiercoles.Text))  
    {  
        txtHastaMiercoles.Text = "hh:mm";  
        txtHastaMiercoles.ForeColor = Color.DimGray;  
    }  
}  
  
//Jueves  
private void txtDesdeJueves_Enter(object sender, EventArgs e)  
{  
    if (txtDesdeJueves.Text == "hh:mm")  
    {  
        txtDesdeJueves.Text = string.Empty;  
        txtDesdeJueves.ForeColor = Color.Black;  
    }  
}  
private void txtDesdeJueves_Leave(object sender, EventArgs e)  
{  
    //Por el lado contrario, cuando se quita el foco  
    //del control, lo volvemos a su estado y color inicial  
  
    if (string.IsNullOrEmpty(txtDesdeJueves.Text))  
    {  
        txtDesdeJueves.Text = "hh:mm";  
        txtDesdeJueves.ForeColor = Color.DimGray;  
    }  
}  
private void txtHastaJueves_Enter(object sender, EventArgs e)  
{  
    if (txtHastaJueves.Text == "hh:mm")  
    {  
        txtHastaJueves.Text = string.Empty;  
        txtHastaJueves.ForeColor = Color.Black;  
    }  
}  
private void txtHastaJueves_Leave(object sender, EventArgs e)  
{  
    //Por el lado contrario, cuando se quita el foco  
    //del control, lo volvemos a su estado y color inicial  
  
    if (string.IsNullOrEmpty(txtHastaJueves.Text))  
    {  
        txtHastaJueves.Text = "hh:mm";  
        txtHastaJueves.ForeColor = Color.DimGray;  
    }  
}  
  
//Viernes  
private void txtDesdeViernes_Enter(object sender, EventArgs e)
```

```
{  
    if (txtDesdeViernes.Text == "hh:mm")  
    {  
        txtDesdeViernes.Text = string.Empty;  
        txtDesdeViernes.ForeColor = Color.Black;  
    }  
}  
}  
private void txtDesdeViernes_Leave(object sender, EventArgs e)  
{  
    //Por el lado contrario, cuando se quita el foco  
    //del control, lo volvemos a su estado y color inicial  
  
    if (string.IsNullOrEmpty(txtDesdeViernes.Text))  
    {  
        txtDesdeViernes.Text = "hh:mm";  
        txtDesdeViernes.ForeColor = Color.DimGray;  
    }  
}  
}  
private void txtHastaViernes_Enter(object sender, EventArgs e)  
{  
    if (txtHastaViernes.Text == "hh:mm")  
    {  
        txtHastaViernes.Text = string.Empty;  
        txtHastaViernes.ForeColor = Color.Black;  
    }  
}  
}  
private void txtHastaViernes_Leave(object sender, EventArgs e)  
{  
    //Por el lado contrario, cuando se quita el foco  
    //del control, lo volvemos a su estado y color inicial  
  
    if (string.IsNullOrEmpty(txtHastaViernes.Text))  
    {  
        txtHastaViernes.Text = "hh:mm";  
        txtHastaViernes.ForeColor = Color.DimGray;  
    }  
}  
}  
  
//Sábado  
private void txtDesdeSabado_Enter(object sender, EventArgs e)  
{  
    if (txtDesdeSabado.Text == "hh:mm")  
    {  
        txtDesdeSabado.Text = string.Empty;  
        txtDesdeSabado.ForeColor = Color.Black;  
    }  
}  
}  
private void txtDesdeSabado_Leave(object sender, EventArgs e)  
{  
    //Por el lado contrario, cuando se quita el foco  
    //del control, lo volvemos a su estado y color inicial  
  
    if (string.IsNullOrEmpty(txtDesdeSabado.Text))  
    {  
        txtDesdeSabado.Text = "hh:mm";  
        txtDesdeSabado.ForeColor = Color.DimGray;  
    }  
}  
}  
private void txtHastaSabado_Enter(object sender, EventArgs e)  
{  
    if (txtHastaSabado.Text == "hh:mm")  
}
```

```

        {
            txtHastaSabado.Text = string.Empty;
            txtHastaSabado.ForeColor = Color.Black;
        }
    }

    private void txtHastaSabado_Leave(object sender, EventArgs e)
    {
        //Por el lado contrario, cuando se quita el foco
        //del control, lo volvemos a su estado y color inicial

        if (string.IsNullOrEmpty(txtHastaSabado.Text))
        {
            txtHastaSabado.Text = "hh:mm";
            txtHastaSabado.ForeColor = Color.DimGray;
        }
    }
}

#endregion

#region Clicks
private void chkTodos_Click(object sender, EventArgs e)
{
    check = 0;
    VerificarChk();
}

private void lblCancelar_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
    this.Close();
}

private void btnAltaJornada_Click(object sender, EventArgs e)
{
    RevisarContenido();
    VerificarFormatoHora();
    if (contenidoIncorrecto)
    {
        MessageBox.Show("Si selecciona un día, el horario correspondiente " +
                        "debe ser registrado de manera correcta.", "Datos erróneos",
                        MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
    }
    else
    {
        JornadasManagement();
        this.Close();
    }
}

#endregion

#region Arrastrar Form

//Se importan archivos DLL para poder realizar la captura de la acción del mouse
//y poder arrastrar los formularios desde donde se realiza la captura

[DllImport("user32.dll", EntryPoint = "ReleaseCapture")]
private extern static void ReleaseCapture();
[DllImport("user32.dll", EntryPoint = "SendMessage")]
private extern static void SendMessage(System.IntPtr hWnd, int wMsg, int wParam, int lParam);
private void Jornadas_MouseDown(object sender, MouseEventArgs e)
{
}

```

```
{  
    ReleaseCapture();  
    SendMessage(this.Handle, 0x112, 0xf012, 0);  
}  
  
#endregion  
  
#region KeyPress en TextBox  
private void txtHastaSabado_KeyPress(object sender, KeyPressEventArgs e)  
{  
    _restricciones.Horarios(e);  
}  
  
private void txtDesdeSabado_KeyPress(object sender, KeyPressEventArgs e)  
{  
    _restricciones.Horarios(e);  
}  
  
private void txtDesdeViernes_KeyPress(object sender, KeyPressEventArgs e)  
{  
    _restricciones.Horarios(e);  
}  
  
private void txtHastaViernes_KeyPress(object sender, KeyPressEventArgs e)  
{  
    _restricciones.Horarios(e);  
}  
  
private void txtDesdeJueves_KeyPress(object sender, KeyPressEventArgs e)  
{  
    _restricciones.Horarios(e);  
}  
  
private void txtHastaJueves_KeyPress(object sender, KeyPressEventArgs e)  
{  
    _restricciones.Horarios(e);  
}  
  
private void txtDesdeMiercoles_KeyPress(object sender, KeyPressEventArgs e)  
{  
    _restricciones.Horarios(e);  
}  
  
private void txtHastaMiercoles_KeyPress(object sender, KeyPressEventArgs e)  
{  
    _restricciones.Horarios(e);  
}  
  
private void txtDesdeMartes_KeyPress(object sender, KeyPressEventArgs e)  
{  
    _restricciones.Horarios(e);  
}  
  
private void txtHastaMartes_KeyPress(object sender, KeyPressEventArgs e)  
{  
    _restricciones.Horarios(e);  
}  
  
private void txtDesdeLunes_KeyPress(object sender, KeyPressEventArgs e)  
{  
    _restricciones.Horarios(e);  
}
```

```
}

private void txtHastaLunes_KeyPress(object sender, KeyPressEventArgs e)
{
    _restricciones.Horarios(e);
}

private string AgregarColon(string texto)
{
    //verificamos si el texto ingresado hasta el momento tiene 2 caracteres
    //si es así, se le agregan los 2 puntos, para evitar ponerlo a cada momento
    //cuando están poniendo los horarios.
    if (texto.Length == 2)
    {
        texto += ":";

    }

    return texto;
}

private void txtDesdeLunes_KeyUp(object sender, KeyEventArgs e)
{
    hora = txtDesdeLunes.Text;
    txtDesdeLunes.Text = AgregarColon(hora);
    txtDesdeLunes.Select(txtDesdeLunes.Text.Length, 0);
}

private void txtHastaLunes_KeyUp(object sender, KeyEventArgs e)
{
    hora = txtHastaLunes.Text;
    txtHastaLunes.Text = AgregarColon(hora);
    txtHastaLunes.Select(txtHastaLunes.Text.Length, 0);
}

private void txtDesdeMartes_KeyUp(object sender, KeyEventArgs e)
{
    hora = txtDesdeMartes.Text;
    txtDesdeMartes.Text = AgregarColon(hora);
    txtDesdeMartes.Select(txtDesdeMartes.Text.Length, 0);
}

private void txtHastaMartes_KeyUp(object sender, KeyEventArgs e)
{
    hora = txtHastaMartes.Text;
    txtHastaMartes.Text = AgregarColon(hora);
    txtHastaMartes.Select(txtHastaMartes.Text.Length, 0);
}

private void txtDesdeMiercoles_KeyUp(object sender, KeyEventArgs e)
{
    hora = txtDesdeMiercoles.Text;
    txtDesdeMiercoles.Text = AgregarColon(hora);
    txtDesdeMiercoles.Select(txtDesdeMiercoles.Text.Length, 0);
}

private void txtHastaMiercoles_KeyUp(object sender, KeyEventArgs e)
{
    hora = txtHastaMiercoles.Text;
    txtHastaMiercoles.Text = AgregarColon(hora);
    txtHastaMiercoles.Select(txtHastaMiercoles.Text.Length, 0);
}
```

```
}

private void txtDesdeJueves_KeyUp(object sender, KeyEventArgs e)
{
    hora = txtDesdeJueves.Text;
    txtDesdeJueves.Text = AgregarColon(hora);
    txtDesdeJueves.Select(txtDesdeJueves.Text.Length, 0);
}

private void txtHastaJueves_KeyUp(object sender, KeyEventArgs e)
{
    hora = txtHastaJueves.Text;
    txtHastaJueves.Text = AgregarColon(hora);
    txtHastaJueves.Select(txtHastaJueves.Text.Length, 0);
}

private void txtDesdeViernes_KeyUp(object sender, KeyEventArgs e)
{
    hora = txtDesdeViernes.Text;
    txtDesdeViernes.Text = AgregarColon(hora);
    txtDesdeViernes.Select(txtDesdeViernes.Text.Length, 0);
}

private void txtHastaViernes_KeyUp(object sender, KeyEventArgs e)
{
    hora = txtHastaViernes.Text;
    txtHastaViernes.Text = AgregarColon(hora);
    txtHastaViernes.Select(txtHastaViernes.Text.Length, 0);
}

private void txtDesdeSabado_KeyUp(object sender, KeyEventArgs e)
{
    hora = txtDesdeSabado.Text;
    txtDesdeSabado.Text = AgregarColon(hora);
    txtDesdeSabado.Select(txtDesdeSabado.Text.Length, 0);
}

private void txtHastaSabado_KeyUp(object sender, KeyEventArgs e)
{
    hora = txtHastaSabado.Text;
    txtHastaSabado.Text = AgregarColon(hora);
    txtHastaSabado.Select(txtHastaSabado.Text.Length, 0);
}

#endregion

#region Selección de checks
private void chkLunes_Click(object sender, EventArgs e)
{
    check = 1;
    VerificarChk();
}
private void chkMartes_Click(object sender, EventArgs e)
{
    check = 2;
    VerificarChk();
    if (chkMartes.Checked && !chkTodos.Checked)
    {
        txtDesdeMartes.Enabled = true;
        txtHastaMartes.Enabled = true;
    }
}
```

```
        else
        {
            txtDesdeMartes.Enabled = false;
            txtHastaMartes.Enabled = false;
        }
    }
private void chkMiercoles_Click(object sender, EventArgs e)
{
    check = 3;
    VerificarChk();
    if (chkMiercoles.Checked && !chkTodos.Checked)
    {
        txtDesdeMiercoles.Enabled = true;
        txtHastaMiercoles.Enabled = true;
    }
    else
    {
        txtDesdeMiercoles.Enabled = false;
        txtHastaMiercoles.Enabled = false;
    }
}
private void chkJueves_Click(object sender, EventArgs e)
{
    check = 4;
    VerificarChk();
    if (chkJueves.Checked && !chkTodos.Checked)
    {
        txtDesdeJueves.Enabled = true;
        txtHastaJueves.Enabled = true;
    }
    else
    {
        txtHastaJueves.Enabled = false;
        txtDesdeJueves.Enabled = false;
    }
}
private void chkViernes_Click(object sender, EventArgs e)
{
    check = 5;
    VerificarChk();
    if (chkViernes.Checked && !chkTodos.Checked)
    {
        txtDesdeViernes.Enabled = true;
        txtHastaViernes.Enabled = true;
    }
    else
    {
        txtDesdeViernes.Enabled = false;
        txtHastaViernes.Enabled = false;
    }
}
private void chkSabado_Click(object sender, EventArgs e)
{
    check = 6;
    VerificarChk();
    if (chkSabado.Checked && !chkTodos.Checked)
    {
        txtDesdeSabado.Enabled = true;
        txtHastaSabado.Enabled = true;
    }
    else
```

```
{  
    txtDesdeSabado.Enabled = false;  
    txtHastaSabado.Enabled = false;  
}  
}  
  
#endregion  
  
#region Click botones de borrado de jornada.  
private void btnDelLunes_Click(object sender, EventArgs e)  
{  
    if (pLunesId != -1)  
    {  
        EliminarJornada(pLunesId);  
        pLunesId = -1;  
    }  
    else if (!string.IsNullOrEmpty(txtDesdeLunes.Text) && txtDesdeLunes.Text !=  
"hh:mm")  
    {  
        txtDesdeLunes.Text = "hh:mm";  
    }  
    else if (!string.IsNullOrEmpty(txtHastaLunes.Text) && txtHastaLunes.Text !=  
"hh:mm")  
    {  
        txtHastaLunes.Text = "hh:mm";  
    }  
    else if (chkTodos.Checked)  
    {  
        chkTodos.Checked = false;  
        todosChk = false;  
    }  
    chkLunes.Checked = false;  
}  
private void btnDelMartes_Click(object sender, EventArgs e)  
{  
    if (pMartesId != -1)  
    {  
        EliminarJornada(pMartesId);  
        pMartesId = -1;  
    }  
    else if (!string.IsNullOrEmpty(txtDesdeMartes.Text) && txtDesdeMartes.Text !=  
"hh:mm")  
    {  
        txtDesdeMartes.Text = "hh:mm";  
    }  
    else if (!string.IsNullOrEmpty(txtHastaMartes.Text) && txtHastaMartes.Text !=  
"hh:mm")  
    {  
        txtHastaMartes.Text = "hh:mm";  
    }  
    else if (chkTodos.Checked)  
    {  
        chkTodos.Checked = false;  
        todosChk = false;  
    }  
    chkMartes.Checked = false;  
}  
private void btnDelMiercoles_Click(object sender, EventArgs e)  
{  
    if (pMiercolesId != -1)  
    {  
        EliminarJornada(pMiercolesId);  
    }  
}
```

```
        pMiercolesId = -1;
    }
    else if (!string.IsNullOrEmpty(txtDesdeMiercoles.Text) && txtDesdeMiercoles.Text
!= "hh:mm")
    {
        txtDesdeMiercoles.Text = "hh:mm";
    }
    else if (!string.IsNullOrEmpty(txtHastaMiercoles.Text) && txtHastaMiercoles.Text
!= "hh:mm")
    {
        txtHastaMiercoles.Text = "hh:mm";
    }
    else if (chkTodos.Checked)
    {
        chkTodos.Checked = false;
        todosChk = false;
    }
    chkMiercoles.Checked = false;
}
private void btnDelJueves_Click(object sender, EventArgs e)
{
    if (pJuevesId != -1)
    {
        EliminarJornada(pJuevesId);
        pJuevesId = -1;
    }
    else if (!string.IsNullOrEmpty(txtDesdeJueves.Text) && txtDesdeJueves.Text !=
"hh:mm")
    {
        txtDesdeJueves.Text = "hh:mm";
    }
    else if (!string.IsNullOrEmpty(txtHastaJueves.Text) && txtHastaJueves.Text !=
"hh:mm")
    {
        txtHastaJueves.Text = "hh:mm";
    }
    else if (chkTodos.Checked)
    {
        chkTodos.Checked = false;
        todosChk = false;
    }
    chkJueves.Checked = false;
}
private void btnDelViernes_Click(object sender, EventArgs e)
{
    if (pViernesId != -1)
    {
        EliminarJornada(pViernesId);
        pViernesId = -1;
    }
    else if (!string.IsNullOrEmpty(txtDesdeViernes.Text) && txtDesdeViernes.Text !=
"hh:mm")
    {
        txtDesdeViernes.Text = "hh:mm";
    }
    else if (!string.IsNullOrEmpty(txtHastaViernes.Text) && txtHastaViernes.Text !=
"hh:mm")
    {
        txtHastaViernes.Text = "hh:mm";
    }
    else if (chkTodos.Checked)
```

```
        chkTodos.Checked = false;
        todosChk = false;
    }
    chkViernes.Checked = false;
}
private void btnDelSabado_Click(object sender, EventArgs e)
{
    if (pSabadoId != -1)
    {
        EliminarJornada(pSabadoId);
        pSabadoId = -1;
    }
    else if (!string.IsNullOrEmpty(txtDesdeSabado.Text) && txtDesdeSabado.Text != "hh:mm")
    {
        txtDesdeSabado.Text = "hh:mm";
    }
    else if (!string.IsNullOrEmpty(txtHastaSabado.Text) && txtHastaSabado.Text != "hh:mm")
    {
        txtHastaSabado.Text = "hh:mm";
    }
    else if (chkTodos.Checked)
    {
        chkTodos.Checked = false;
        todosChk = false;
    }
    chkSabado.Checked = false;
}

#endregion

}
}
```

```
using BussinessLayer;
using Entities;
using System;
using System.Data;
using System.Drawing;
using System.Windows.Forms;

namespace Gym
{
    public partial class Pagos : Form
    {
        #region Instancias
        //negocio
        private readonly BussinessPersonas _bussinesPersonas;
        private readonly BussinessCaja _bussinessCaja;
        private readonly BussinessPlanes _bussinessPlanes;

        //entidades
        private readonly Personas _personas;
        private readonly Detalles_Cajas _detalles_Cajas;
        private readonly Entities.Cajas _caja;
        private readonly Entities.Planes _planes;

        //clases internas
        private readonly Restricciones _restricciones;

        #endregion

        #region Variables

        private int personaLogueada;
        private int cliente_ID;
        private int idCajaAbierta;
        private string buscar;
        private DataSet DsClienteDatos;
        private DataSet DsDetalle;
        private DataTable DtPlanesCliente;
        private bool datosVacios = false;
        private bool camposVacios;
        private bool rbSeleccionado;
        private bool cajaAbierta;

        #endregion

        public Pagos(int idPersonaLogin)
        {
            InitializeComponent();

            personaLogueada = idPersonaLogin;

            _restricciones = new Restricciones();

            _bussinessCaja = new BussinessCaja();
            _bussinessPlanes = new BussinessPlanes();
            _bussinesPersonas = new BussinessPersonas();

            _personas = new Personas();
            _caja = new Entities.Cajas();
            _planes = new Entities.Planes();
        }
    }
}
```

```
_detalles_Cajas = new Detalles_Cajas();

VerificarCajaAbierta();
CargarDetalles();
}

#region Metodos

private void CargarDetalles()
{
    dtgvDetalles.Rows.Clear();
    DsDetalle = _bussinessCaja.GetDetalles(buscar);
    buscar = string.Empty;

    if (DsDetalle.Tables[0].Rows.Count > 0)
    {
        foreach (DataRow dr in DsDetalle.Tables[0].Rows)
        {
            dtgvDetalles.Rows.Add(dr[0].ToString(), dr[1], dr[2], dr[3]);
        }
    }
}

private void VerificarCajaAbierta()
{
    DateTime fecha = DateTime.Now;
    cajaAbierta = _bussinessCaja.VerificarCajaAbierta(fecha);

    if (cajaAbierta)
    {
        GetLastCajaID();
    }
}
private void GetLastCajaID()
{
    _bussinessCaja.GetLastCajaID(_caja);
    idCajaAbierta = _caja.Caja_ID;
}

private void MostrarCliente()
{
    DsClienteDatos = _bussinesPersonas.GetPersonaPlan(buscar, _personas);
    AcomodarDatos();
}

private void AcomodarDatos()
{
    if (DsClienteDatos.Tables[0].Rows.Count > 0)
    {
        foreach (DataRow dr in DsClienteDatos.Tables[0].Rows)
        {
            cliente_ID = int.Parse(dr[1].ToString());
            lblNombre.Text += dr[2].ToString() + " " + dr[3].ToString();
            lblDocumento.Text += dr[4].ToString();
            lblTelefono.Text += dr[5].ToString();
            lblMail.Text += dr[6].ToString();
            break;
        }
        camposVacios = false;
    }
}
```

```
        }

    else
    {
        MessageBox.Show("No hay clientes con el documento ingresado. " +
            "Por favor, intente de nuevo, o haga primero el registro del cliente.",
            "Datos no encontrados");
        camposVacios = true;
    }
}

private void BuscarPlanesParaPagoAlumnos()
{
    DtPlanesCliente = _bussinessPlanes.GetPlanesParaPago(cliente_ID);
    cmbPlanesPagaPago.DataSource = DtPlanesCliente;
    cmbPlanesPagaPago.DisplayMember = "Nombre";
    cmbPlanesPagaPago.ValueMember = "Plan_ID";
}

private void CostoPlan(string idPlan)
{
    int id = Convert.ToInt32(idPlan);
    _planes.Plan_ID = id;
    _bussinessPlanes.GetCostoPlan(_planes);
    txtImporte.Text = "$" + _planes.Importe_Plan.ToString();
    txtImporte.ForeColor = Color.Black;
}

private void ResetControlsCliente()
{
    lblNombre.Text = "Nombre: ";
    lblDocumento.Text = "DNI: ";
    lblTelefono.Text = "Telefono: ";
    lblMail.Text = "Mail: ";
    lblPlanesAsignadosCliente.Text = string.Empty;
}

#endregion

#region Enter Controls
private void TxtDatosClienteSaldos_Enter(object sender, System.EventArgs e)
{
    if (txtBuscarCliente.Text == "DNI")
    {
        txtBuscarCliente.Text = string.Empty;
        txtBuscarCliente.ForeColor = Color.Black;
    }
}

private void TxtDatosClienteSaldos_Leave(object sender, System.EventArgs e)
{
    if (string.IsNullOrEmpty(txtBuscarCliente.Text))
    {
        txtBuscarCliente.Text = "DNI";
        txtBuscarCliente.ForeColor = Color.DimGray;
    }
}

private void TxtImporte_Enter(object sender, EventArgs e)
{
    if (txtImporte.Text == "Importe $")
    {
        txtImporte.Text = string.Empty;
```

```
        txtImporte.ForeColor = Color.Black;
    }

    private void TxtImporte_Leave(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(txtImporte.Text))
    {
        txtImporte.Text = "Importe $";
        txtImporte.ForeColor = Color.DimGray;
    }
}

private void txtBuscarMovimiento_Enter(object sender, EventArgs e)
{
    if (txtBuscarMovimiento.Text == "Buscar Movimientos")
    {
        txtBuscarMovimiento.Text = string.Empty;
        txtBuscarMovimiento.ForeColor = Color.Black;
    }
}

private void txtBuscarMovimiento_Leave(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(txtBuscarMovimiento.Text))
    {
        txtBuscarMovimiento.Text = "Buscar Movimientos";
        txtBuscarMovimiento.ForeColor = Color.DimGray;
    }
}

private void txtObservaciones_Enter(object sender, EventArgs e)
{
    if (txtObservaciones.Text == "Observaciones")
    {
        txtObservaciones.Text = string.Empty;
        txtObservaciones.ForeColor = Color.Black;
    }
}

private void txtObservaciones_Leave(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(txtObservaciones.Text))
    {
        txtObservaciones.Text = "Observaciones";
        txtObservaciones.ForeColor = Color.DimGray;
    }
}

#endregion

#region Eventos
private void txtBuscarCliente_KeyPress(object sender, KeyPressEventArgs e)
{
    buscar = txtBuscarCliente.Text;
    _restricciones.SoloNumeros(e, buscar);

    if (e.KeyChar == Convert.ToChar(Keys.Enter))
    {
        if (!datosVacios)
        {
            if (string.IsNullOrEmpty(buscar))
```

```

        MessageBox.Show("Tiene que buscar con un número de documento válido.
No se adminten campos vacíos.",
                    "Gestión en proceso", MessageBoxButtons.OKCancel);
    }
    else
    {
        ResetControlsCliente();
        MostrarCliente();
        BuscarPlanesParaPagoAlumnos();
        datosVacios = true;
    }
}
else
{
    DialogResult result = MessageBox.Show("Tiene un cliente cargado. ¿Quiere
interrumpir la gestión y buscar un cliente nuevo?",
                    "Gestión en proceso", MessageBoxButtons.OKCancel);
    if (result == DialogResult.OK)
    {
        if (string.IsNullOrEmpty(buscar))
        {
            MessageBox.Show("Tiene que buscar con un número de documento
válido. No se adminten campos vacíos.",
                    "Gestión en proceso", MessageBoxButtons.OKCancel);
        }
        else
        {
            ResetControlsCliente();
            MostrarCliente();
            buscar = string.Empty;
        }
    }
}
}

private void BtnRegistrarMovimiento_Click(object sender, EventArgs e)
{
    if (cajaAbierta)
    {
        //normalizamos el importe porque tiene el signo $
        string strImporte = txtImporte.Text;
        strImporte = strImporte.Substring(1, strImporte.Length - 1);
        decimal importe = Convert.ToDecimal(strImporte);

        //traemos el nombre
        string nombre = lblNombre.Text;
        nombre = nombre.Substring(8, nombre.Length - 8);

        string tipoMovimiento;

        if (rbCobro.Checked) tipoMovimiento = "Cobro";
        else tipoMovimiento = "Pago";

        DialogResult result = MessageBox.Show($"Va a registrar un {tipoMovimiento} de
${importe}, para el alumno {nombre}.\n" +
                    $"¿{Es correcto?}" , "Registro de Movimientos", MessageBoxButtons.OKCancel,
                    MessageBoxIcon.Question);
        if (result == DialogResult.OK)
        {
    }
}

```

```

        _detalles_Cajas.Caja_ID = idCajaAbierta;
        _detalles_Cajas.Empleado_ID = personaLogueada;
        _detalles_Cajas.Observaciones = Convert.ToString(txtObservaciones.Text);

        if (rbCobro.Checked)
        {
            _detalles_Cajas.Importe_Ingreso = importe;
            _detalles_Cajas.Plan_Asignado_ID =
Convert.ToInt32(cmbPlanesPagaPago.SelectedValue);
            _bussinessCaja.RegistrarCobro(_detalles_Cajas);
        }
        else
        {
            _detalles_Cajas.Importe_Egreso = Convert.ToDecimal(txtImporte.Text);
            _bussinessCaja.RegistrarPago(_detalles_Cajas);
        }

        MessageBox.Show($"Se registró correctamente el {tipoMovimiento} de
${importe}.",
                    "Registro de Movimientos", MessageBoxButtons.OKCancel,
MessageBoxIcon.Information);

        CargarDetalles();
    }
}
else
{
    MessageBox.Show("Primero tiene que abrir la caja, antes de registrar
movimientos.", "Caja cerrada", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}

#endregion

private void rbCobro_CheckedChanged(object sender, EventArgs e)
{
    if (rbCobro.Checked)
    {
        cmbPlanesPagaPago.Enabled = true;
    }
    else
    {
        cmbPlanesPagaPago.Enabled = false;
        txtImporte.Text = "Importe $";
        txtImporte.ForeColor = Color.DimGray;
    }
}

private void cmbPlanesPagaPago_SelectionChangeCommitted(object sender, EventArgs e)
{
    string idPlan = Convert.ToString(cmbPlanesPagaPago.SelectedValue);
    CostoPlan(idPlan);
}

private void txtBuscarClase_KeyPress(object sender, KeyPressEventArgs e)
{
    buscar = txtBuscarMovimiento.Text;
    if (e.KeyChar == Convert.ToChar(Keys.Enter))
    {
        CargarDetalles();
    }
}

```

```
    }  
}  
}
```

```
using BussinessLayer;
using Entities;
using System;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;

namespace Gym
{
    public partial class Planes : Form
    {
        #region Instancias
        //Capa de negocio
        private readonly BussinessPersonas _bussinesPersonas;
        private readonly BussinessPlanes _bussinessPlanes;
        private readonly BussinessClientes _bussinesClientes;
        private readonly BussinessAsistencia _bussinessAsistencia;
        private readonly BussinessPlanesAsignados _bussinesPlanesAsignados;
        private readonly BussinessJornadas _bussinessJornadas;

        //Entities
        private readonly Entities.Clientes _clientes;
        private readonly Entities.Planes _planes;
        private readonly Entities.Asistencias _asistencias;
        private readonly Entities.Planes_Asignados _planesAsignados;
        private readonly Jornadas_Planes _jornadasPlanes;
        private Personas _personas;

        //Clases internas
        private readonly MetodosGenerales _metodosGenerales;
        private readonly Restricciones _restricciones;

        #endregion

        #region Variables
        private string desde;
        private string hasta;
        private int personaLogueada;
        private string buscar;
        private DataSet DsClienteDatos;
        private DataSet DsPlanesAsignados;
        private DataSet DsPlanes;
        private DataTable DtPlanesAsignados;
        private DataTable DtJornadaDePlan;
        private DataTable DtProfesores;
        private DataTable DtPlanes;
        private bool camposVacios = false;
        private bool datosVacios = false;
        private int cliente_ID;
        private int planSeleccionado;
        private int contador;
        private bool todosChk;
        private int pTodosId = -1;
        private int pLunesId = -1;
        private int pMartesId = -1;
        private int pMiercolesId = -1;
        private int pJuevesId = -1;
        private int pViernesId = -1;
        private int pSabadoId = -1;
        
```

```

private int idJornada;
private int check;
private bool esAlta = true;
private bool cargarJornada;
private int idPlanAEditar;
private string hora;
private bool contenidoIncorrecto = false;
private bool duplicidad = false;

#endregion

#region Loading
public Planes(int idPersonaLog)
{
    InitializeComponent();

    _bussinessPlanes = new BussinessPlanes();
    _bussinesClientes = new BussinessClientes();
    _bussinesPersonas = new BussinessPersonas();
    _bussinessJornadas = new BussinessJornadas();
    _bussinessAsistencia = new BussinessAsistencia();
    _planesAsignados = new Entities.Planes_Asignados();
    _bussinesPlanesAsignados = new BussinessPlanesAsignados();

    _personas = new Personas();
    _planes = new Entities.Planes();
    _clientes = new Entities.Clientes();
    _jornadasPlanes = new Jornadas_Planes();
    _asistencias = new Entities.Asistencias();

    _restricciones = new Restricciones();
    _metodosGenerales = new MetodosGenerales();

    personaLogueada = idPersonaLog;
    btnAsignarPlan.Enabled = false;

    BuscarPlanes();
    BuscarProfesores();
    CargarPlanes();
}
#endregion

#region Encapsulamiento

private void CargarPlanes()
{
    dtgvPlanes.Rows.Clear();
    DsPlanes = _bussinessPlanes.GetPlanesActuales(buscar);
    buscar = string.Empty;

    if (DsPlanes.Tables[0].Rows.Count > 0)
    {
        foreach (DataRow dr in DsPlanes.Tables[0].Rows)
        {
            dtgvPlanes.Rows.Add(dr[0].ToString(), dr[1], dr[2], dr[3]);
        }
    }
}

```

```
private void BuscarProfesores()
{
    DtProfesores = _bussinesPersonas.BuscaProfesores(_personas);
    cmbProfesores.DataSource = DtProfesores;
    cmbProfesores.DisplayMember = "Nombre";
    cmbProfesores.ValueMember = "Empleado_ID";
}

private void BuscarPlanes()
{
    DtPlanes = _bussinessPlanes.GetPlanes(_planes);
    cmbPlanesActivos.DataSource = DtPlanes;
    cmbPlanesActivos.DisplayMember = "Nombre";
    cmbPlanesActivos.ValueMember = "Plan_ID";
}

private void AcomodarDatos()
{
    if (DsClienteDatos.Tables[0].Rows.Count > 0)
    {
        foreach (DataRow dr in DsClienteDatos.Tables[0].Rows)
        {
            cliente_ID = int.Parse(dr[1].ToString());
            lblNombre.Text += dr[2].ToString() + " " + dr[3].ToString();
            lblDocumento.Text += dr[4].ToString();
            lblTelefono.Text += dr[5].ToString();
            lblMail.Text += dr[6].ToString();
            break;
        }
    }
    else
    {
        MessageBox.Show("No hay clientes con el documento ingresado. " +
                        "Por favor, intente de nuevo, o haga primero el registro del cliente.",
                        "Datos no encontrados");
    }
}
private void MostrarCliente()
{
    DsClienteDatos = _bussinesPersonas.GetPersonaPlan(buscar, _personas);
    AcomodarDatos();
    DsPlanesAsignados = _bussinesPlanesAsignados.BuscarPlanesAsignados(cliente_ID);
    AcomodarPlanesAsignados();
    btnAsignarPlan.Enabled = true;
}
private void AcomodarPlanesAsignados()
{
    int i = 0;
    foreach (DataRow dr in DsPlanesAsignados.Tables[0].Rows)
    {
        lblPlanesAsignadosCliente.Text += dr[0].ToString();
        i++;
        if ((i + 1) <= DsPlanesAsignados.Tables[0].Rows.Count)
        {
            lblPlanesAsignadosCliente.Text += ", ";
        }
    }
}

private void AsigarlePlanAlCliente()
{
```

```

_planesAsignados.Plan_ID = Convert.ToInt32(cmbPlanesActivos.SelectedValue);
_planesAsignados.Empleado_ID = personaLogueada;
_planesAsignados.Cliente_ID = cliente_ID;
_planesAsignados.Fecha_Inscripcion = DateTime.Now;
_planesAsignados.Estado = "A";
_bussinesPlanesAsignados.AsginarPlanAlCliente(_planesAsignados);

//Acá hacemos la resta de cupos.
_planes.Plan_ID = Convert.ToInt32(cmbPlanesActivos.SelectedValue);
_bussinessPlanes.EditarCupoRestante(_planes);
}

private void RevisarCamposVacios()
{
    foreach (Control ctrl in gpDatosPlan.Controls)
    {
        if (ctrl is TextBox box)
        {
            TextBox t;
            t = box;
            if (string.IsNullOrEmpty(t.Text))
            {
                camposVacios = true;
                break;
            }
        }
        else
        {
            camposVacios = false;
        }
    }
}

private void RegistrarNuevoPlan()
{
    _planes.Persona_ID = personaLogueada;
    _planes.Nombre = txtNombrePlan.Text;
    _planes.Importe_Plan = Convert.ToDecimal(txtCostoMensual.Text);
    _planes.Empleado_ID = Convert.ToInt32(cmbProfesores.SelectedValue);
    _planes.Duracion = Convert.ToInt32(txtDuracion.Text);
    _planes.Cupo_Total = Convert.ToInt32(txtCupoTotal.Text);
    _planes.Cupo_Restante = Convert.ToInt32(txtCupoTotal.Text);
    _planes.Fecha_Inicio = dtpFechaInicio.Value;
    _planes.Fecha_Alta_Plan = DateTime.Now;
    _planes.Estado = "A";
    _bussinessPlanes.RegistrarNuevoPlan(_planes);
}

private void AsignacionHoras()
{
    int contadorDesde = desde.Length;
    int contadorHasta = hasta.Length;

    if (contadorDesde <= 5 && contadorHasta <= 5)
    {
        _jornadasPlanes.Desde_Hora = _metodosGenerales.ConvertirHoraAfecha(desde);
        _jornadasPlanes.Hasta_Hora = _metodosGenerales.ConvertirHoraAfecha(hasta);
    }
    else
    {
        MessageBox.Show("El campo de horas, está mal registrado. Desde registrarse con");
    }
}

```

```
este formato: ##:##", "Formato incorrecto", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }
    }

    private void EditarPlanes()
    {
        _planes.Plan_ID = idPlanAEitar;
        _planes.Nombre = txtNombrePlan.Text;
        _planes.Importe_Plan = Convert.ToDecimal(txtCostoMensual.Text);
        _planes.Duracion = Convert.ToInt32(txtDuracion.Text);
        _planes.Cupo_Total = Convert.ToInt32(txtCupoTotal.Text);
        _planes.Fecha_Inicio = dtpFechaInicio.Value;
        _planes.Empleado_ID = Convert.ToInt32(cmbProfesores.SelectedValue);

        _bussinessPlanes.EditarPlan(_planes);
    }

    private void GuardarJornada()
    {
        foreach (Control chk in this.gbJornadaPlanes.Controls)
        {
            if (chk is CheckBox box)
            {
                CheckBox c;
                c = box;
                //Y luego de cada checkbox le asignamos el horario.
                //Lunes
                if (c.Checked)
                {
                    switch (c.Text)
                    {
                        case "Lunes":
                            _jornadasPlanes.Dia = "Lunes";
                            _jornadasPlanes.Jornada_Plan_ID = pLunesId;
                            desde = txtDesdeLunes.Text.ToString();
                            hasta = txtHastaLunes.Text.ToString();
                            AsignacionHoras();
                            break;
                        case "Martes":
                            _jornadasPlanes.Dia = "Martes";
                            _jornadasPlanes.Jornada_Plan_ID = pMartesId;
                            desde = txtDesdeMartes.Text.ToString();
                            hasta = txtHastaMartes.Text.ToString();
                            AsignacionHoras();
                            break;
                        case "Miercoles":
                            _jornadasPlanes.Dia = "Miercoles";
                            _jornadasPlanes.Jornada_Plan_ID = pMiercolesId;
                            desde = txtDesdeMiercoles.Text.ToString();
                            hasta = txtHastaMiercoles.Text.ToString();
                            AsignacionHoras();
                            break;
                        case "Jueves":
                            _jornadasPlanes.Dia = "Jueves";
                            _jornadasPlanes.Jornada_Plan_ID = pJuevesId;
                            desde = txtDesdeJueves.Text.ToString();
                            hasta = txtHastaJueves.Text.ToString();
                            AsignacionHoras();
                            break;
                        case "Viernes":
                            _jornadasPlanes.Dia = "Viernes";
                            break;
                    }
                }
            }
        }
    }
```

```

                _jornadasPlanes.Jornada_Plan_ID = pViernesId;
                desde = txtDesdeViernes.Text.ToString();
                hasta = txtHastaViernes.Text.ToString();
                AsignacionHoras();
                break;
            case "Sabado":
                _jornadasPlanes.Dia = "Sabado";
                _jornadasPlanes.Jornada_Plan_ID = pSabadoId;
                desde = txtDesdeSabado.Text.ToString();
                hasta = txtHastaSabado.Text.ToString();
                AsignacionHoras();
                break;
        }
        _jornadasPlanes.Esto = "A";
        _jornadasPlanes.Plan_ID = idJornada;
        if (_jornadasPlanes.Plan_ID != -1)
        {
            _bussinessJornadas.EditarJornadaPlan(_jornadasPlanes);
        }
        else
        {
            _bussinessJornadas.AltaJornadaPlan(_jornadasPlanes);
        }
        desde = string.Empty;
        hasta = string.Empty;
    }
}
}

private void JornadasManagement()
{
    //Acá editamos
    if (cargarJornada)
    {
        GuardarJornada();
    }
    else
    {
        //Acá creamos:

        //Ahora tenemos 2 opciones.
        // 1 es que se haya seleccionado el check "todos"
        if (todosChk)
        {
            _jornadasPlanes.Plan_ID = idJornada;
            _jornadasPlanes.Dia = "Todos";
            _jornadasPlanes.Esto = "A";
            desde = txtDesdeLunes.Text.ToString();
            hasta = txtHastaLunes.Text.ToString();
            AsignacionHoras();

            _bussinessJornadas.AltaJornadaPlan(_jornadasPlanes);
        }
        else
        {
            //Y la otra es que se hayan seleccionado varios días
            //Vamos a verificar los checkbox que se hayan seleccionado
            GuardarJornada();
        }
    }
}

```

```
}

private void GetJornadaDePlan()
{
    bool esEmpleado = false;
    bool darBaja = false;
    int plan_ID = Convert.ToInt32(cmbPlanesActivos.SelectedValue);
    Jornadas jn = new Jornadas(plan_ID, esEmpleado, darBaja);
    jn.ShowDialog();
}

private void RevisarTodosChk()
{
    foreach (Control chk in gbJornadaPlanes.Controls)
    {
        if (chk is CheckBox box)
        {
            CheckBox c;
            c = box;
            if (c.Checked)
            {
                contador++;
            }
        }
    }
}

private void ChkDias()
{
    RevisarTodosChk();
    if (chkTodos.Checked)
    {
        chkTodos.Checked = false;
        todosChk = false;
    }
    else
    {
        if (contador > 5 && !chkTodos.Checked)
        {
            ActivarTodoChk();
        }
    }
}

private void ActivarTodoChk()
{
    if (!chkTodos.Checked && contador < 7 && !todosChk)
    {
        foreach (Control chk in gbJornadaPlanes.Controls)
        {
            if (chk is CheckBox box)
            {
                CheckBox c;
                c = box;
                c.Checked = true;
            }
        }
        todosChk = true;
    }
    else
    {
        if (todosChk)
        {
            //Si el chk que dice "Seleccionar todos" está seleccionado
        }
    }
}
```

```

//Entonces todos los otros chk se van a seleccionar
//Caso contrario, se les quitará el chkeck
foreach (Control chk in this.gbJornadaPlanes.Controls)
{
    if (chk is CheckBox box)
    {
        CheckBox c;
        c = box;
        //Deshabilito el check en cada checkbox
        c.Checked = false;
    }
}
todosChk = false;
}
else
{
    foreach (Control chk in this.gbJornadaPlanes.Controls)
    {
        if (chk is CheckBox box)
        {
            CheckBox c;
            c = box;
            c.Checked = true;
        }
    }
    todosChk = true;
}
}

private void VerificarChk()
{
    switch (check)
    {
        case 0:
            ActivarTodoChk();
            break;
        case 1:
            ChkDias();
            break;
        case 2:
            ChkDias();
            break;
        case 3:
            ChkDias();
            break;
        case 4:
            ChkDias();
            break;
        case 5:
            ChkDias();
            break;
        case 6:
            ChkDias();
            break;
    }
    contador = 0;
}
private void ResetControls()
{
    lblCostoMensual.Text = "Costo mensual: $";
    lblCuposRestantes.Text = "0";
}

```

```
lblCuposTotales.Text = "0";
}

private void BuscarDatosPlan()
{
    ResetControls();
    _planes.Plan_ID = Convert.ToInt32(cmbPlanesActivos.SelectedValue);
    _bussinessPlanes.GetDatoPlan(_planes);
    planSeleccionado = _planes.Plan_ID;
    lblCostoMensual.Text += _planes.Importe_Plan.ToString();
    lblCuposRestantes.Text = _planes.Cupo_Restante.ToString();
    lblCuposTotales.Text = _planes.Cupo_Total.ToString();
}

private void ResetControlsCliente()
{
    lblNombre.Text = "Nombre: ";
    lblDocumento.Text = "DNI: ";
    lblTelefono.Text = "Telefono: ";
    lblMail.Text = "Mail: ";
    lblPlanesAsignadosCliente.Text = string.Empty;
}

private void ResetControlsAltaPlan()
{
    foreach (Control ctrl in gpDatosPlan.Controls)
    {
        if (ctrl is TextBox box)
        {
            TextBox txt;
            txt = box;
            txt.Text = string.Empty;
        }

        if (ctrl is CheckBox chk)
        {
            CheckBox c;
            c = chk;
            c.Checked = false;
        }
    }
    foreach (Control ctrl in gbJornadaPlanes.Controls)
    {
        if (ctrl is TextBox box)
        {
            TextBox txt;
            txt = box;
            txt.Text = "hh:mm";
            txt.ForeColor = Color.DimGray;
        }

        if (ctrl is CheckBox chk)
        {
            CheckBox c;
            c = chk;
            c.Checked = false;
        }
    }
}

private void ComprobarDuplicidadDePlan()
{
```

```

        int idplan = Convert.ToInt32(cmbPlanesActivos.SelectedValue);
        duplicidad = _bussinesPlanesAsignados.BuscarDuplicidad(idplan, cliente_ID);
    }

#endregion

#region Eventos Keypress
private void txtBuscarClase_KeyPress(object sender, KeyPressEventArgs e)
{
    buscar = txtBuscarClase.Text;
    if (e.KeyChar == Convert.ToChar(Keys.Enter))
    {
        CargarPlanes();
    }
}
private void txtBuscarCliente_KeyPress(object sender, KeyPressEventArgs e)
{
    buscar = txtBuscarCliente.Text;
    _restricciones.SoloNumeros(e, buscar);
    if (e.KeyChar == Convert.ToChar(Keys.Enter))
    {
        if (!datosVacios)
        {
            if (string.IsNullOrEmpty(buscar))
            {
                MessageBox.Show("Tiene que buscar con un número de documento válido.
No se adminten campos vacíos.",
                    "Gestión en proceso", MessageBoxButtons.OKCancel);
            }
            else
            {
                ResetControlsCliente();
                MostrarCliente();
                datosVacios = true;
            }
        }
        else
        {
            DialogResult result = MessageBox.Show("Tiene un cliente cargado. ¿Quiere
interrumpir la gestión y buscar un cliente nuevo?",
                "Gestión en proceso", MessageBoxButtons.OKCancel);
            if (result == DialogResult.OK)
            {
                if (string.IsNullOrEmpty(buscar))
                {
                    MessageBox.Show("Tiene que buscar con un número de documento
válido. No se adminten campos vacíos.",
                        "Gestión en proceso", MessageBoxButtons.OKCancel);
                }
                else
                {
                    ResetControlsCliente();
                    MostrarCliente();
                    buscar = string.Empty;
                }
            }
        }
    }
}

#endregion

```

```

#region Alta de Planes

private void btnAltaPlan_Click(object sender, EventArgs e)
{
    RevisarCamposVacios();
    VerificarFormatoHora();
    if (camposVacios)
    {
        MessageBox.Show("Para registrar un nuevo plan, no deben haber campos vacíos",
            "Campos incompletos");
    }
    else
    {
        if (esAlta)
        {
            RegistrarNuevoPlan();
            _bussinessPlanes.GetLastID(_planes);
            idPlanAEeditar = _planes.Plan_ID;
        }
        else
        {
            EditarPlanes();
        }

        JornadasManagement();
        MessageBox.Show("El plan se ha guardado con éxito!",
            "Registro de Planes");

        //Una vez dado de alta, actualizamos la lista.
        CargarPlanes();
        ResetControlsAltaPlan();
        esAlta = true;
    }
}

private void VerificarFormatoHora()
{
    foreach (Control ctrl in gbJornadaPlanes.Controls)
    {
        if (ctrl is TextBox box)
        {
            TextBox t;
            t = box;
            if (t.Text.Length < 5)
            {
                contenidoIncorrecto = true; ;
                break;
            }
            else
            {
                contenidoIncorrecto = false;
            }
        }
    }
}

#endregion

#region Focus Textbox
private void txtBuscarCliente_Enter(object sender, EventArgs e)

```

```

{
    if (txtBuscarCliente.Text == "DNI")
    {
        txtBuscarCliente.Text = string.Empty;
        txtBuscarCliente.ForeColor = Color.Black;
    }
}

private void txtBuscarCliente_Leave(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(txtBuscarCliente.Text))
    {
        txtBuscarCliente.Text = "DNI";
        txtBuscarCliente.ForeColor = Color.DimGray;
    }
}

private void txtBuscarClase_Enter(object sender, EventArgs e)
{
    if (txtBuscarClase.Text == "Buscar Clase")
    {
        txtBuscarClase.Text = string.Empty;
        txtBuscarClase.ForeColor = Color.Black;
    }
}

private void txtBuscarClase_Leave(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(txtBuscarClase.Text))
    {
        txtBuscarClase.Text = "Buscar Clase";
        txtBuscarClase.ForeColor = Color.DimGray;
    }
}

#endregion

#region Asignación de planes a un cliente
private void btnAsignarPlan_Click(object sender, EventArgs e)
{
    if (camposVacios)
    {
        MessageBox.Show("No hay cliente seleccionado para asignarle un plan." +
            "Por favor, busque el cliente primero, y luego asigne el plan.");
    }
    else
    {
        ComprobarDuplicidadDePlan();
        if (duplicidad)
        {
            MessageBox.Show("Este plan ya está asignado al cliente. Por favor, seleccione otro plan", "Duplicidad encontrada");
        }
        else
        {
            if (Convert.ToInt32(lblCuposRestantes.Text) == 0 &&
Convert.ToInt32(lblCuposTotales.Text) > 0)
            {
                MessageBox.Show("No hay cupos disponibles para esta clase.");
            }
        }
    }
}

```

```

        {
            //Asignarle el plan.
            AsigarlePlanAlCliente();

            //Actualizar planes
            cmbPlanesActivos_SelectionChangeCommitted(sender, e);

            string lblnombre = lblNombre.Text;
            string nombre = lblnombre.Substring(8, lblnombre.Length - 8);
            string plan = Convert.ToString(cmbPlanesActivos.Text);

            MessageBox.Show($"Se asignó correctamente el plan: {plan}, para el
alumno: {nombre}", "Asignación de Planes");
        }
    }
}

private void lblVerJornadas_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
    if (cmbProfesores.Items.Count > 0)
    {
        //traemos la jornada del plan seleccionado.
        GetJornadaDePlan();
    }
    else
    {
        MessageBox.Show("Debe seleccionar primero un plan para poder buscar los
horarios.", "Horarios de Planes", MessageBoxButtons.OK);
    }
}

private void cmbPlanesActivos_SelectionChangeCommitted(object sender, EventArgs e)
{
    //Me busca el precio, cupo total y restante.
    BuscarDatosPlan();
    //Busca los horarios
}

#endregion

#region Selección de checks
private void chkTodos_Click(object sender, EventArgs e)
{
    check = 0;
    VerificarChk();
}
private void chkLunes_Click(object sender, EventArgs e)
{
    check = 1;
    VerificarChk();
}
private void chkMartes_Click(object sender, EventArgs e)
{
    check = 2;
    VerificarChk();
    if (chkMartes.Checked && !chkTodos.Checked)
    {
        txtDesdeMartes.Enabled = true;
        txtHastaMartes.Enabled = true;
    }
    else
    {

```

```
        txtDesdeMartes.Enabled = false;
        txtHastaMartes.Enabled = false;
    }
}
private void chkMiercoles_Click(object sender, EventArgs e)
{
    check = 3;
    VerificarChk();
    if (chkMiercoles.Checked && !chkTodos.Checked)
    {
        txtDesdeMiercoles.Enabled = true;
        txtHastaMiercoles.Enabled = true;
    }
    else
    {
        txtDesdeMiercoles.Enabled = false;
        txtHastaMiercoles.Enabled = false;
    }
}
private void chkJueves_Click(object sender, EventArgs e)
{
    check = 4;
    VerificarChk();
    if (chkJueves.Checked && !chkTodos.Checked)
    {
        txtDesdeJueves.Enabled = true;
        txtHastaJueves.Enabled = true;
    }
    else
    {
        txtHastaJueves.Enabled = false;
        txtDesdeJueves.Enabled = false;
    }
}
private void chkViernes_Click(object sender, EventArgs e)
{
    check = 5;
    VerificarChk();
    if (chkViernes.Checked && !chkTodos.Checked)
    {
        txtDesdeViernes.Enabled = true;
        txtHastaViernes.Enabled = true;
    }
    else
    {
        txtDesdeViernes.Enabled = false;
        txtHastaViernes.Enabled = false;
    }
}
private void chkSabado_Click(object sender, EventArgs e)
{
    check = 6;
    VerificarChk();
    if (chkSabado.Checked && !chkTodos.Checked)
    {
        txtDesdeSabado.Enabled = true;
        txtHastaSabado.Enabled = true;
    }
    else
    {
        txtDesdeSabado.Enabled = false;
```

```
        txtHastaSabado.Enabled = false;
    }
}

#endregion

#region TextBox de Hora

//Lunes
private void txtDesdeLunes_Enter(object sender, EventArgs e)
{
    if (txtDesdeLunes.Text == "hh:mm")
    {
        txtDesdeLunes.Text = string.Empty;
        txtDesdeLunes.ForeColor = Color.Black;
    }
}
private void txtDesdeLunes_Leave(object sender, EventArgs e)
{
    //Por el lado contrario, cuando se quita el foco
    //del control, lo volvemos a su estado y color inicial

    if (string.IsNullOrEmpty(txtDesdeLunes.Text))
    {
        txtDesdeLunes.Text = "hh:mm";
        txtDesdeLunes.ForeColor = Color.DimGray;
    }
}
private void txtHastaLunes_Enter(object sender, EventArgs e)
{
    if (txtHastaLunes.Text == "hh:mm")
    {
        txtHastaLunes.Text = string.Empty;
        txtHastaLunes.ForeColor = Color.Black;
    }
}
private void txtHastaLunes_Leave(object sender, EventArgs e)
{
    //Por el lado contrario, cuando se quita el foco
    //del control, lo volvemos a su estado y color inicial

    if (string.IsNullOrEmpty(txtHastaLunes.Text))
    {
        txtHastaLunes.Text = "hh:mm";
        txtHastaLunes.ForeColor = Color.DimGray;
    }
}

//Martes
private void txtDesdeMartes_Enter(object sender, EventArgs e)
{
    if (txtDesdeMartes.Text == "hh:mm")
    {
        txtDesdeMartes.Text = string.Empty;
        txtDesdeMartes.ForeColor = Color.Black;
    }
}
private void txtDesdeMartes_Leave(object sender, EventArgs e)
{
    //Por el lado contrario, cuando se quita el foco
    //del control, lo volvemos a su estado y color inicial
```

```
if (string.IsNullOrEmpty(txtDesdeMartes.Text))
{
    txtDesdeMartes.Text = "hh:mm";
    txtDesdeMartes.ForeColor = Color.DimGray;
}
}
private void txtHastaMartes_Enter(object sender, EventArgs e)
{
    if (txtHastaMartes.Text == "hh:mm")
    {
        txtHastaMartes.Text = string.Empty;
        txtHastaMartes.ForeColor = Color.Black;
    }
}
private void txtHastaMartes_Leave(object sender, EventArgs e)
{
    //Por el lado contrario, cuando se quita el foco
    //del control, lo volvemos a su estado y color inicial

    if (string.IsNullOrEmpty(txtHastaMartes.Text))
    {
        txtHastaMartes.Text = "hh:mm";
        txtHastaMartes.ForeColor = Color.DimGray;
    }
}

//Miércoles
private void txtDesdeMiercoles_Enter(object sender, EventArgs e)
{
    if (txtDesdeMiercoles.Text == "hh:mm")
    {
        txtDesdeMiercoles.Text = string.Empty;
        txtDesdeMiercoles.ForeColor = Color.Black;
    }
}
private void txtDesdeMiercoles_Leave(object sender, EventArgs e)
{
    //Por el lado contrario, cuando se quita el foco
    //del control, lo volvemos a su estado y color inicial

    if (string.IsNullOrEmpty(txtDesdeMiercoles.Text))
    {
        txtDesdeMiercoles.Text = "hh:mm";
        txtDesdeMiercoles.ForeColor = Color.DimGray;
    }
}
private void txtHastaMiercoles_Enter(object sender, EventArgs e)
{
    if (txtHastaMiercoles.Text == "hh:mm")
    {
        txtHastaMiercoles.Text = string.Empty;
        txtHastaMiercoles.ForeColor = Color.Black;
    }
}
private void txtHastaMiercoles_Leave(object sender, EventArgs e)
{
    //Por el lado contrario, cuando se quita el foco
    //del control, lo volvemos a su estado y color inicial

    if (string.IsNullOrEmpty(txtHastaMiercoles.Text))
```

```
{  
    txtHastaMiercoles.Text = "hh:mm";  
    txtHastaMiercoles.ForeColor = Color.DimGray;  
}  
}  
  
//Jueves  
private void txtDesdeJueves_Enter(object sender, EventArgs e)  
{  
    if (txtDesdeJueves.Text == "hh:mm")  
    {  
        txtDesdeJueves.Text = string.Empty;  
        txtDesdeJueves.ForeColor = Color.Black;  
    }  
}  
}  
private void txtDesdeJueves_Leave(object sender, EventArgs e)  
{  
    //Por el lado contrario, cuando se quita el foco  
    //del control, lo volvemos a su estado y color inicial  
  
    if (string.IsNullOrEmpty(txtDesdeJueves.Text))  
    {  
        txtDesdeJueves.Text = "hh:mm";  
        txtDesdeJueves.ForeColor = Color.DimGray;  
    }  
}  
private void txtHastaJueves_Enter(object sender, EventArgs e)  
{  
    if (txtHastaJueves.Text == "hh:mm")  
    {  
        txtHastaJueves.Text = string.Empty;  
        txtHastaJueves.ForeColor = Color.Black;  
    }  
}  
private void txtHastaJueves_Leave(object sender, EventArgs e)  
{  
    //Por el lado contrario, cuando se quita el foco  
    //del control, lo volvemos a su estado y color inicial  
  
    if (string.IsNullOrEmpty(txtHastaJueves.Text))  
    {  
        txtHastaJueves.Text = "hh:mm";  
        txtHastaJueves.ForeColor = Color.DimGray;  
    }  
}  
  
//Viernes  
private void txtDesdeViernes_Enter(object sender, EventArgs e)  
{  
    if (txtDesdeViernes.Text == "hh:mm")  
    {  
        txtDesdeViernes.Text = string.Empty;  
        txtDesdeViernes.ForeColor = Color.Black;  
    }  
}  
private void txtDesdeViernes_Leave(object sender, EventArgs e)  
{  
    //Por el lado contrario, cuando se quita el foco  
    //del control, lo volvemos a su estado y color inicial  
  
    if (string.IsNullOrEmpty(txtDesdeViernes.Text))
```

```
{  
    txtDesdeViernes.Text = "hh:mm";  
    txtDesdeViernes.ForeColor = Color.DimGray;  
}  
}  
private void txtHastaViernes_Enter(object sender, EventArgs e)  
{  
    if (txtHastaViernes.Text == "hh:mm")  
    {  
        txtHastaViernes.Text = string.Empty;  
        txtHastaViernes.ForeColor = Color.Black;  
    }  
}  
private void txtHastaViernes_Leave(object sender, EventArgs e)  
{  
    //Por el lado contrario, cuando se quita el foco  
    //del control, lo volvemos a su estado y color inicial  
  
    if (string.IsNullOrEmpty(txtHastaViernes.Text))  
    {  
        txtHastaViernes.Text = "hh:mm";  
        txtHastaViernes.ForeColor = Color.DimGray;  
    }  
}  
  
//Sábado  
private void txtDesdeSabado_Enter(object sender, EventArgs e)  
{  
    if (txtDesdeSabado.Text == "hh:mm")  
    {  
        txtDesdeSabado.Text = string.Empty;  
        txtDesdeSabado.ForeColor = Color.Black;  
    }  
}  
private void txtDesdeSabado_Leave(object sender, EventArgs e)  
{  
    //Por el lado contrario, cuando se quita el foco  
    //del control, lo volvemos a su estado y color inicial  
  
    if (string.IsNullOrEmpty(txtDesdeSabado.Text))  
    {  
        txtDesdeSabado.Text = "hh:mm";  
        txtDesdeSabado.ForeColor = Color.DimGray;  
    }  
}  
private void txtHastaSabado_Enter(object sender, EventArgs e)  
{  
    if (txtHastaSabado.Text == "hh:mm")  
    {  
        txtHastaSabado.Text = string.Empty;  
        txtHastaSabado.ForeColor = Color.Black;  
    }  
}  
private void txtHastaSabado_Leave(object sender, EventArgs e)  
{  
    //Por el lado contrario, cuando se quita el foco  
    //del control, lo volvemos a su estado y color inicial  
  
    if (string.IsNullOrEmpty(txtHastaSabado.Text))  
    {  
        txtHastaSabado.Text = "hh:mm";  
    }  
}
```

```
        txtHastaSabado.ForeColor = Color.DimGray;
    }

}

private string AgregarColon(string texto)
{
    //verificamos si el texto ingresado hasta el momento tiene 2 caracteres
    //si es así, se le agregan los 2 puntos, para evitar ponerlo a cada momento
    //cuando están poniendo los horarios.
    if (texto.Length == 2)
    {
        texto += ":";

    }

    return texto;
}

private void txtDesdeLunes_KeyUp(object sender, KeyEventArgs e)
{
    hora = txtDesdeLunes.Text;
    txtDesdeLunes.Text = AgregarColon(hora);
    txtDesdeLunes.Select(txtDesdeLunes.Text.Length, 0);
}

private void txtHastaLunes_KeyUp(object sender, KeyEventArgs e)
{
    hora = txtHastaLunes.Text;
    txtHastaLunes.Text = AgregarColon(hora);
    txtHastaLunes.Select(txtHastaLunes.Text.Length, 0);
}

private void txtDesdeMartes_KeyUp(object sender, KeyEventArgs e)
{
    hora = txtDesdeMartes.Text;
    txtDesdeMartes.Text = AgregarColon(hora);
    txtDesdeMartes.Select(txtDesdeMartes.Text.Length, 0);
}

private void txtHastaMartes_KeyUp(object sender, KeyEventArgs e)
{
    hora = txtHastaMartes.Text;
    txtHastaMartes.Text = AgregarColon(hora);
    txtHastaMartes.Select(txtHastaMartes.Text.Length, 0);
}

private void txtDesdeMiercoles_KeyUp(object sender, KeyEventArgs e)
{
    hora = txtDesdeMiercoles.Text;
    txtDesdeMiercoles.Text = AgregarColon(hora);
    txtDesdeMiercoles.Select(txtDesdeMiercoles.Text.Length, 0);
}

private void txtHastaMiercoles_KeyUp(object sender, KeyEventArgs e)
{
    hora = txtHastaMiercoles.Text;
    txtHastaMiercoles.Text = AgregarColon(hora);
    txtHastaMiercoles.Select(txtHastaMiercoles.Text.Length, 0);
}

private void txtDesdeJueves_KeyUp(object sender, KeyEventArgs e)
```

```
{  
    hora = txtDesdeJueves.Text;  
    txtDesdeJueves.Text = AgregarColon(hora);  
    txtDesdeJueves.Select(txtDesdeJueves.Text.Length, 0);  
}  
  
private void txtHastaJueves_KeyUp(object sender, KeyEventArgs e)  
{  
    hora = txtHastaJueves.Text;  
    txtHastaJueves.Text = AgregarColon(hora);  
    txtHastaJueves.Select(txtHastaJueves.Text.Length, 0);  
}  
  
private void txtDesdeViernes_KeyUp(object sender, KeyEventArgs e)  
{  
    hora = txtDesdeViernes.Text;  
    txtDesdeViernes.Text = AgregarColon(hora);  
    txtDesdeViernes.Select(txtDesdeViernes.Text.Length, 0);  
}  
  
private void txtHastaViernes_KeyUp(object sender, KeyEventArgs e)  
{  
    hora = txtHastaViernes.Text;  
    txtHastaViernes.Text = AgregarColon(hora);  
    txtHastaViernes.Select(txtHastaViernes.Text.Length, 0);  
}  
  
private void txtDesdeSabado_KeyUp(object sender, KeyEventArgs e)  
{  
    hora = txtDesdeSabado.Text;  
    txtDesdeSabado.Text = AgregarColon(hora);  
    txtDesdeSabado.Select(txtDesdeSabado.Text.Length, 0);  
}  
  
private void txtHastaSabado_KeyUp(object sender, KeyEventArgs e)  
{  
    hora = txtHastaSabado.Text;  
    txtHastaSabado.Text = AgregarColon(hora);  
    txtHastaSabado.Select(txtHastaSabado.Text.Length, 0);  
}  
  
private void txtDesdeLunes_KeyPress(object sender, KeyPressEventArgs e)  
{  
    _restricciones.Horarios(e);  
}  
  
private void txtHastaLunes_KeyPress(object sender, KeyPressEventArgs e)  
{  
    _restricciones.Horarios(e);  
}  
  
private void txtDesdeMartes_KeyPress(object sender, KeyPressEventArgs e)  
{  
    _restricciones.Horarios(e);  
}  
  
private void txtHastaMartes_KeyPress(object sender, KeyPressEventArgs e)  
{  
    _restricciones.Horarios(e);  
}
```

```

private void txtDesdeMiercoles_KeyPress(object sender, KeyPressEventArgs e)
{
    _restricciones.Horarios(e);
}

private void txtHastaMiercoles_KeyPress(object sender, KeyPressEventArgs e)
{
    _restricciones.Horarios(e);
}

private void txtDesdeJueves_KeyPress(object sender, KeyPressEventArgs e)
{
    _restricciones.Horarios(e);
}

private void txtHastaJueves_KeyPress(object sender, KeyPressEventArgs e)
{
    _restricciones.Horarios(e);
}

private void txtDesdeViernes_KeyPress(object sender, KeyPressEventArgs e)
{
    _restricciones.Horarios(e);
}

private void txtHastaViernes_KeyPress(object sender, KeyPressEventArgs e)
{
    _restricciones.Horarios(e);
}

private void txtDesdeSabado_KeyPress(object sender, KeyPressEventArgs e)
{
    _restricciones.Horarios(e);
}

private void txtHastaSabado_KeyPress(object sender, KeyPressEventArgs e)
{
    _restricciones.Horarios(e);
}

#endregion

#region Eventos de botones
private void btnEliminarPlanes_Click(object sender, EventArgs e)
{
    string lim = lblPlanesAsignadosCliente.Text;
    EditarPlanesDeCliente ep = new EditarPlanesDeCliente(lim);
    ep.ShowDialog();
}

private void btnEditarPlan_Click(object sender, EventArgs e)
{
    esAlta = false;
    idPlanAEellar = Convert.ToInt32(dtgvPlanes.CurrentRow.Cells[0].Value);
    CargarPlanParaEdicion();
    GetJornadaPlan();
}

private void CargarPlanParaEdicion()
{
    //traigo el plan con el ID
}

```

```

_planes.Plan_ID = Convert.ToInt32(dtgvPlanes.CurrentRow.Cells[0].Value);
_bussinessPlanes.GetPlanUnico(_planes);

//Vamos a asignar los datos del plan en cada cuadro de texto.
idPlanAEEditar = _planes.Plan_ID;
txtNombrePlan.Text = _planes.Nombre.ToString();
txtCostoMensual.Text = _planes.Importe_Plan.ToString();
txtDuracion.Text = _planes.Duracion.ToString();
txtCupoTotal.Text = _planes.Cupo_Total.ToString();
dtpFechaInicio.Value = _planes.Fecha_Inicio;
cmbProfesores.SelectedValue = _planes.Empleado_ID;
}

private void GetJornadaPlan()
{
    _jornadasPlanes.Plan_ID = idPlanAEEditar;
    //Una vez asignado, busco la jornada
    //Y la asigno a un datatable, para poder manejar el contenido.
    DtJornadaDePlan = _bussinessJornadas.GetJornadaPlan(_jornadasPlanes);

    if (DtJornadaDePlan.Rows.Count > 0)
    {
        cargarJornada = true;
        //cargo la jornada en los textbox correspondientes
        CargarJornada();
    }
    else
    {
        cargarJornada = false;
    }
}

private void CargarJornada()
{
    foreach (DataRow dr in DtJornadaDePlan.Rows)
    {
        //Vamos a switchear cada vuelta para que vaya
        //colocando los datos cada vuelta donde corresponda.
        switch (dr[2].ToString())
        {
            case "Todos":
                chkTodos.Checked = true;
                //Este es el Id de la jornada que se carga
                pTodosId = Convert.ToInt32(dr[0]);
                txtDesdeLunes.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[3].ToString()));
                txtHastaLunes.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[4].ToString()));
                txtDesdeLunes.ForeColor = Color.Black;
                txtHastaLunes.ForeColor = Color.Black;
                VerificarChk();
                break;
            case "Lunes":
                txtDesdeLunes.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[3].ToString()));
                Convert.ToString(dr[3]);
                txtHastaLunes.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[4].ToString()));
                txtDesdeLunes.ForeColor = Color.Black;
                txtHastaLunes.ForeColor = Color.Black;
                pLunesId = Convert.ToInt32(dr[0]);
        }
    }
}

```

```

        chkLunes.Checked = true;
        break;
    case "Martes":
        txtDesdeMartes.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[3].ToString()));
        txtHastaMartes.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[4].ToString()));
        txtDesdeMartes.ForeColor = Color.Black;
        txtHastaMartes.ForeColor = Color.Black;
        pMartesId = Convert.ToInt32(dr[0]);
        chkMartes.Checked = true;
        break;
    case "Miércoles":
        txtDesdeMiércoles.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[3].ToString()));
        txtHastaMiércoles.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[4].ToString()));
        txtDesdeMiércoles.ForeColor = Color.Black;
        txtHastaMiércoles.ForeColor = Color.Black;
        pMiércolesId = Convert.ToInt32(dr[0]);
        chkMiércoles.Checked = true;
        break;
    case "Jueves":
        txtDesdeJueves.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[3].ToString()));
        txtHastaJueves.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[4].ToString()));
        txtDesdeJueves.ForeColor = Color.Black;
        txtHastaJueves.ForeColor = Color.Black;
        pJuevesId = Convert.ToInt32(dr[0]);
        chkJueves.Checked = true;
        break;
    case "Viernes":
        txtDesdeViernes.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[3].ToString()));
        txtHastaViernes.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[4].ToString()));
        txtDesdeViernes.ForeColor = Color.Black;
        txtHastaViernes.ForeColor = Color.Black;
        pViernesId = Convert.ToInt32(dr[0]);
        chkViernes.Checked = true;
        break;
    case "Sábado":
        txtDesdeSabado.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[3].ToString()));
        txtHastaSabado.Text =
Convert.ToString(_metodosGenerales.ConvertirfechaAhora(dr[4].ToString()));
        txtDesdeSabado.ForeColor = Color.Black;
        txtHastaSabado.ForeColor = Color.Black;
        pSabadoId = Convert.ToInt32(dr[0]);
        chkSabado.Checked = true;
        break;
    }
}
}

private void btnEliminarPlan_Click(object sender, EventArgs e)
{
    DialogResult result = MessageBox.Show("¿Está seguro de querer eliminar el plan? Si es así, presione 'Aceptar'", "Atención! Eliminación de Plan", MessageBoxButtons.OKCancel);
    if (result == DialogResult.OK)
    {

```

```
_planes.Plan_ID = Convert.ToInt32(dtgvPlanes.CurrentRow.Cells[0].Value);
_bussinessPlanes.EliminarPlan(_planes);

MessageBox.Show("¿Está seguro de querer eliminar el plan?",
"Atención! Eliminación de Plan", MessageBoxButtons.OKCancel);
}

}

#endregion
```

```
}
```

```
}
```

```
using BussinessLayer;
using Entities;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;

namespace Gym
{
    public class MetodosGenerales
    {
        #region Instancias

        //Entidades
        private Personas _personas;
        private Entities.Empleados _empleados;
        private Jornadas_Empleados _jornadas_Empleados;

        //Capa de negocio
        private BussinessTipo _bussinessTipo;
        private BussinessEmpleados _bussinessEmpleados;
        private readonly BussinessPersonas _bussinessPersonas;
        private readonly Tipos_Empleados _tiposEmpleados;
        private readonly BussinessJornadas _bussinessJornadas;

        #endregion

        #region Ctor
        public MetodosGenerales()
        {
            _bussinessTipo = new BussinessTipo();
            _bussinessEmpleados = new BussinessEmpleados();
            _empleados = new Entities.Empleados();
            _bussinessPersonas = new BussinessPersonas();
            _personas = new Personas();
            _tiposEmpleados = new Tipos_Empleados();
            _jornadas_Empleados = new Jornadas_Empleados();
            _bussinessJornadas = new BussinessJornadas();
        }
        #endregion

        #region Variables
        //Esta variable me sirve para saber el id del empleado que tiene abierta
        //su sesión en el programa
        public int usuarioOpenID;
        public int personaOpenID;
        public string nombrePersona;
        public bool CargarJornada = true;
        public int empleadoID;
        public int personaID;

        //resto de variables
        public bool CajaAbierta = true;
        public DataTable DtTipos_Documentos = new DataTable();
        public DataTable DtTipos_Sexos = new DataTable();
        public DataTable DtTipos_Empleados = new DataTable();
        public DataTable DtEstados_Empleados = new DataTable();
    }
}
```

```
public int empleado_ID;
public int persona_ID;
public string apellidoNombrePersona;
public string tipoNumDocumentoPersona;
public string estadoPersonaPersona;

#endregion

#region Planes

#endregion

#region Tipos de documento

// Traeremos los tipos de documento
public void Bring_Tipos_Documentos()
{
    DtTipos_Documentos = _bussinessTipo.BringTipoDocumento();
}

#endregion

#region Tipos de sexo

// Traeremos los tipos de sexo
public void Bring_Tipos_Sexos()
{
    //Asignamos los dato de ese método de negocios
    //a una table que hemos creado al principio
    DtTipos_Sexos = _bussinessTipo.BringTipoSexo();
}

#endregion

#region Tipos de Empleados

public void Bring_Tipos_Empleados()
{
    //Asignamos los dato de ese método de negocios
    //a una table que hemos creado al principio
    DtTipos_Empleados = _bussinessTipo.BringTipoEmpleado();
}

#endregion

#region Personas y Empleados

public void AltaPersona(string Nombre,
                        string Apellido,
                        int Tipo_Documento_ID,
                        string Nro_documento,
                        int Tipo_Sexo_ID,
                        string Nro_Telefono,
                        string Nro_Alternativo,
                        string Mail,
                        string Observaciones,
                        DateTime FechaAlta)
{
    _personas.Nombre = Nombre;
    _personas.Apellido = Apellido;
    _personas.Tipo_Documento_ID = Tipo_Documento_ID;
}
```

```
_personas.Nro_documento = Nro_documento;
_personas.Tipo_Sexo_ID = Tipo_Sexo_ID;
_personas.Nro_Telefono = Nro_Telefono;
_personas.Nro_Alternativo = Nro_Alternativo;
_personas.Mail = Mail;
_personas.Observaciones = Observaciones;
_personas.Fecha_Alta = FechaAlta;
_bussinessPersonas.AltaPersona(_personas);

//Una vez registrada esta nueva persona, consulto su ID
//Ya que el sistema lo necesita para, dar de alta al empleado
//o al cliente.

GetLastID();
}

public void AltaJefe(string Nombre,
                     string Apellido,
                     int Tipo_Documento_ID,
                     string Nro_documento,
                     int Tipo_Sexo_ID,
                     string Nro_Telefono,
                     string Mail,
                     DateTime FechaAlta)
{
    _personas.Nombre = Nombre;
    _personas.Apellido = Apellido;
    _personas.Tipo_Documento_ID = Tipo_Documento_ID;
    _personas.Nro_documento = Nro_documento;
    _personas.Tipo_Sexo_ID = Tipo_Sexo_ID;
    _personas.Nro_Telefono = Nro_Telefono;
    _personas.Mail = Mail;
    _personas.Fecha_Alta = FechaAlta;

    //Una vez registrada esta nueva persona, consulto su ID
    //Ya que el sistema lo necesita para, dar de alta al empleado
    //o al cliente.

    GetLastID();
}

public void GetLastID()
{
    _personas = _bussinessPersonas.Get_Last_Id_Persona(_personas);
    persona_ID = _personas.Persona_ID;
}

public void EditarPersona(int Persona_ID, string Nombre,
                         string Apellido,
                         int Tipo_Documento_ID,
                         string Nro_documento,
                         int Tipo_Sexo_ID,
                         string Nro_Telefono,
                         string Nro_Alternativo,
                         string Mail,
                         string Observaciones)
{
    _personas.Persona_ID = Persona_ID;
    _personas.Nombre = Nombre;
    _personas.Apellido = Apellido;
    _personas.Tipo_Documento_ID = Tipo_Documento_ID;
```

```

    _personas.Nro_documento = Nro_documento;
    _personas.Tipo_Sexo_ID = Tipo_Sexo_ID;
    _personas.Nro_Teléfono = Nro_Teléfono;
    _personas.Nro_Alternativo = Nro_Alternativo;
    _personas.Mail = Mail;
    _personas.Observaciones = Observaciones;
    _bussinessPersonas.EditarPersona(_personas);
}

public void Get_Last_Id_Empleado(int personaJor)
{
    _empleados = _bussinessEmpleados.GetLastID(personaJor, _empleados);
    empleado_ID = _empleados.Empleado_ID;
}

public void GetPersona()
{
    _personas = _bussinessPersonas.GetPersona(_personas);
    nombrePersona = _personas.Nombre;
    personaOpenID = _personas.Persona_ID;
}

public void GetEstadosEmpleados()
{
    DtEstados_Empleados = _bussinessEmpleados.GetEstadosEmpleados(_tiposEmpleados);
}

#endregion

#region Conversión de horarios

public String ConvertirfechaHora(string fecha)
{
    string fechaString = fecha;
    string horaString;

    string shora = fechaString.Substring(11, 2);
    string sminutos = sminutos = fechaString.Substring(14, 2);

    horaString = $"{shora}:{sminutos}";
    return horaString;
}

public DateTime ConvertirHoraAfecha(string horario)
{
    string fechaString = "2020-10-27";
    string horaString = horario;

    //Extraer año del fechaString
    string syear = fechaString.Substring(0, 4);
    //Extraer mes del fechaString
    string smonth = fechaString.Substring(5, 2);
    //Extraer dia del fechaString
    string sday = fechaString.Substring(8, 2);
    //Estraer horas
    string shora = horaString.Substring(0, 2);
    //Estraer minutos
    string sminuto = horaString.Substring(3, 2);

    //Convertir a int
    int year, month, day, hora, minuto;
}

```

```
int.TryParse(syear, out year);
int.TryParse(smonth, out month);
int.TryParse(sday, out day);
int.TryParse(shora, out hora);
int.TryParse(sminuto, out minuto);

DateTime fechaConHora = new DateTime(year, month, day, hora, minuto, 0);

return fechaConHora;
}

#endregion
}
```

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

namespace Gym
{
    public class Restricciones
    {
        //Vamos a encapsular las restricciones de cada textbox
        //para que sean comunes a todas las plantillas y solo
        //se acceda a la clase.

        #region Variables globales

        #endregion

        //Solo se pueden ingresar números
        public void SoloNumeros(KeyPressEventArgs e, string strTexto)
        {
            //Solo se teclean los dígitos
            if (Char.IsDigit(e.KeyChar))
            {
                e.Handled = false;
            }

            //permitir teclas de control como retroceso
            else if (Char.IsControl(e.KeyChar))
            {
                e.Handled = false;
            }
            else
            {
                //con esto se desactivan todas las otras teclas no contempladas en las líneas
                e.Handled = true;
            }
        }

        public void Horarios(KeyPressEventArgs e)
        {
            //Solo se teclean los dígitos, los : y las teclas de retroceso, o controles.
            if (Char.IsDigit(e.KeyChar) || Char.IsControl(e.KeyChar) || e.KeyChar == 58 || e.KeyChar == (char)Keys.Back)
            {
                e.Handled = false;
            }

            else
            {
                e.Handled = true;
            }
        }
    }
}
```