```
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 enparticular, 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 GetAlumosDeLaClase()
        {
            //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 uqe 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 recoorido 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 uqe tienen tilde por las
uqe 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 recoorido 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 uqe tienen tilde por las
uqe no)
                    //entonces va a entrar en este if.
                    if (strDato[i] == letras[j])
                    {
                        //para hacer que esa letra sea reemplazada por su vocal sin tilde.
                        strDato = strDato.Replace(strDato[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 strDato;
        }
        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.GetDatoPlan(_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
adminten 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());
            GetAlumosDeLaClase();
            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);
            }
            GetAlumosDeLaClase();
```

```
        }
        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
                {
                    //Asignarle 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.Estado = "A";
            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();
        }
    }
}
```