```
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)
            {
                //nromalizamos 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();
            }
```

```
            }

        }
    }
```