TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TIJUANA

SUBDIRECCIÓN ACADÉMICA

DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

SEMESTRE:

Enero-Junio 2025

CARRERA:

INGENIERÍA EN SISTEMAS COMPUTACIONALES

MATERIA Y SERIE :

INGENIERÍA DE SOFTWARE

TÍTULO:

LENGUAJE DEL SISTEMA “TIJUANA LEGALIS”

UNIDAD A EVALUAR:

4TA UNIDAD

NOMBRES Y NÚMEROS DE CONTROL DE LOS ALUMNOS:

ZARZA MORALES JOSE DIEGO 22210366

ROJAS OSUNA JORDHY JAIR 22210349

SEBASTIAN MARTINEZ GARCIA 22211608

NOMBRE DEL MAESTRO:

MARTHA ELENA PULIDO

FECHA:

13 de mayo del 2025

Índice

Introducción

El proyecto consiste en el desarrollo de un sistema de gestión de casos para bufetes de abogados denominado “Tijuana Legalis”. Este sistema busca optimizar la administración de casos legales, permitiendo un control eficiente de los registros de clientes, casos y abogados, así como la asignación adecuada de recursos legales. La implementación de este software responde a la necesidad de modernizar los procesos administrativos en el ámbito jurídico, mejorando la organización, seguridad de la información y accesibilidad a datos relevantes.

Antecedentes

La gestión tradicional de casos legales en bufetes de abogados suele depender de métodos manuales o sistemas poco integrados, lo que genera ineficiencias, errores y dificultades para rastrear información crítica. Con el aumento de la complejidad en el ámbito jurídico, surge la necesidad de soluciones tecnológicas que automaticen tareas, optimicen la asignación de casos y garanticen la confidencialidad de los datos. Este proyecto se basa en la premisa de que un software especializado puede resolver estas problemáticas, ofreciendo una plataforma centralizada y eficiente.

Planteamiento

El problema principal que aborda el proyecto es la gestión ineficiente de casos y la asignación de abogados en un bufete. La falta de un sistema organizado dificulta el control de datos como la información del cliente, el estado del caso y las competencias de los abogados asignados. Esto puede derivar en retrasos, errores administrativos y una distribución inadecuada de recursos. La solución propuesta es un sistema de gestión que automatice el registro, seguimiento y asignación de casos, optimizando los procesos operativos del bufete.

Objetivos

Objetivo General

Mejorar la gestión de un bufete de abogados mediante un sistema que optimice el manejo de casos y la asignación de recursos, garantizando un orden eficiente en las tareas.

Objetivos Específicos

- Gestionar de manera eficiente los procesos de un bufete de abogados.
- Optimizar los modelos tradicionales de organización de casos.
- Generar opciones efectivas para la distribución de casos entre abogados.
- Organizar mejor los casos asignados a los abogados.
- Mejorar la seguridad de la información manejada por el bufete.

Justificación

La adopción de un software especializado para la gestión de casos legales se justifica por la creciente complejidad del ámbito jurídico. Este sistema ofrecerá una plataforma centralizada para registrar y seguir casos, mejorando la eficiencia y precisión en la gestión. La necesidad de acceso rápido a información crucial, generación de informes detallados y cumplimiento de plazos legales respalda la implementación. Además, la automatización de tareas administrativas facilitará la colaboración entre el equipo legal, permitiendo un uso más estratégico de los recursos y el tiempo.

Resumen del Proyecto

“Tijuana Legalis” es un sistema diseñado para profesionales del derecho que permite registrar y gestionar casos legales, clientes y abogados de manera eficiente. Incluye funcionalidades como registro de datos, seguimiento de plazos, asignación de casos y generación de informes. La interfaz es intuitiva, con cinco pantallas principales para créditos, registro de clientes, abogados, casos y visualización de información. El sistema utiliza Microsoft SQL Server como gestor de base de datos y está desarrollado para operar en entornos Windows, garantizando compatibilidad y seguridad.

Tamaño del Proyecto

El proyecto es de alcance mediano, diseñado para satisfacer las necesidades de bufetes de abogados de pequeño a mediano tamaño. Involucra el desarrollo de una aplicación de escritorio con una base de datos relacional, cinco interfaces principales y funcionalidades específicas para la gestión de casos. Se estima que el sistema manejará hasta cientos de registros de casos, clientes y abogados, con capacidad de escalabilidad para adaptarse a un crecimiento futuro.

Materiales a Necesitar

- **Software**:
  - Sistema operativo: Windows 10 o Windows 11.
  - Sistema gestor de base de datos: Microsoft SQL Server.
  - Entorno de desarrollo: Visual Studio (para programación en C#).
- **Hardware**:
  - Computadora con procesador de 1.6 GHz o superior (x86 o AMD64/x64).
  - 1 GB de RAM.
  - 1 GB de espacio disponible en disco duro.
  - Resolución de pantalla de 1024x768 o superior.
- **Recursos Humanos**:
  - Desarrolladores (2): Jose Diego Zarza Morales y Jordhy Jair Rojas Osuna.
  - Documentador: Sebastian Martinez Garcia.
  - Supervisor: Martha Elena Pulido (maestra).

Tiempo Requerido para la Creación del Proyecto

El desarrollo del proyecto se llevó a cabo durante el semestre de enero a junio de 2025, con un plazo aproximado de 4 meses. La entrega final está programada para el 27 de marzo de 2025. Las fases incluyeron:

- Planificación y análisis: 3 semanas.
- Diseño de interfaces y base de datos: 4 semanas.
- Codificación: 6 semanas.
- Pruebas y ajustes: 3 semanas.
- Documentación y entrega: 2 semanas.

Localización del Proyecto

El proyecto fue desarrollado en el Instituto Tecnológico de Tijuana, dentro del Departamento de Sistemas y Computación, como parte de la asignatura de Ingeniería de Software para la carrera de Ingeniería en Sistemas Computacionales. Las actividades se realizaron de manera presencial y remota, utilizando herramientas de colaboración y desarrollo disponibles en el instituto y en entornos personales de los estudiantes.

Ingeniería del Proyecto

El proyecto sigue un enfoque de ingeniería de software estructurado, dividido en fases de análisis, diseño, implementación, pruebas y documentación. Se diseñó una base de datos relacional con tablas para Clientes, Procuradores y Asuntos, conectadas mediante claves primarias y foráneas. El sistema utiliza procedimientos almacenados para operaciones como inserción, actualización y eliminación de datos. La arquitectura del software es cliente-servidor, con una interfaz de usuario desarrollada en Windows Forms y lógica de negocio implementada en C#.

Consideraciones

Consideración de Software

- **Sistema Operativo**: Windows 10 o Windows 11, asegurando compatibilidad con entornos comunes en bufetes.
- **Sistema Gestor de Base de Datos**: Microsoft SQL Server, seleccionado por su robustez, soporte para grandes volúmenes de datos y facilidad de integración con aplicaciones .NET.
- **Lenguaje de Programación**: C#, utilizado para el desarrollo de la lógica del sistema y la interfaz de usuario en Windows Forms.
- **Herramientas de Desarrollo**: Visual Studio, que proporciona un entorno integrado para codificación, depuración y pruebas.
- **Requisitos No Funcionales**:
  - Rendimiento: Capacidad para manejar grandes volúmenes de datos con tiempos de respuesta rápidos.
  - Fiabilidad: Minimización de errores y garantía de integridad de datos.
  - Seguridad: Cifrado de datos y control de acceso para proteger información confidencial.
  - Escalabilidad: Adaptación al crecimiento en usuarios y datos.

Consideración de Hardware

- **Requisitos Mínimos**:
  - Procesador de 1.6 GHz o superior.
  - 1 GB de RAM.
  - 1 GB de espacio en disco duro.
  - Resolución de pantalla de 1024x768.
- **Dispositivos Compatibles**: Computadoras de escritorio o portátiles con sistemas operativos Windows compatibles.

Consideraciones Adicionales

No se requieren drivers adicionales, y el sistema está optimizado para hardware estándar, garantizando accesibilidad para bufetes con recursos limitados.

Organigrama de Actividades

1. **Análisis y Planificación**: Definición de requisitos funcionales y no funcionales, identificación del público objetivo y planteamiento del problema.
2. **Diseño**: Creación del modelo de base de datos, diseño de interfaces y prototipos.
3. **Desarrollo**: Codificación de las funcionalidades, implementación de la base de datos y procedimientos almacenados.
4. **Pruebas**: Validación de la funcionalidad, corrección de errores y pruebas de usabilidad.
5. **Documentación**: Elaboración del informe final, incluyendo código, manual técnico y referencias.
6. **Entrega**: Presentación del sistema y documentación al supervisor del proyecto.

Prototipo

El prototipo consiste en una aplicación de escritorio con una interfaz gráfica que permite la gestión de clientes, abogados y casos legales.

Plan Rápido/Modelo de Diseño Rápido

Se adoptó un modelo de diseño rápido basado en el enfoque iterativo, con ciclos cortos de desarrollo y retroalimentación:

1. **Requisitos Iniciales**: Identificación de necesidades básicas del bufete.
2. **Prototipo Rápido**: Creación de interfaces básicas para pruebas de usabilidad.
3. **Iteraciones**: Ajustes basados en retroalimentación de usuarios simulados (estudiantes y profesores).
4. **Diseño Final**: Implementación completa de funcionalidades y optimización de la interfaz.

Este enfoque permitió validar el diseño y funcionalidad en etapas tempranas, reduciendo riesgos.

Evaluación del proyecto

Factibilidad Económica

El proyecto no implica costos significativos, ya que se desarrolló con herramientas de software gratuitas o disponibles en el instituto (Visual Studio Community, SQL Server Express). Los únicos recursos necesarios fueron las computadoras personales de los desarrolladores y el tiempo invertido. Para un bufete, el sistema es económicamente viable, ya que opera en hardware estándar y no requiere licencias costosas.

Factibilidad Operativa

El sistema es operativamente factible, ya que está diseñado para usuarios con conocimientos básicos de informática, como abogados y personal administrativo. La interfaz intuitiva y los tutoriales incluidos reducen la curva de aprendizaje. Además, el sistema es compatible con los procesos existentes en un bufete, integrándose fácilmente en flujos de trabajo legales.

Riesgos

- **Riesgos Técnicos**: Posibles errores en la integración de la base de datos o fallos en la interfaz durante el uso intensivo. Mitigación: Pruebas exhaustivas y validación de datos.
- **Riesgos de Usabilidad**: Dificultades para usuarios no familiarizados con tecnología. Mitigación: Diseño intuitivo y soporte contextual.
- **Riesgos de Seguridad**: Acceso no autorizado a datos sensibles. Mitigación: Implementación de cifrado y autenticación de usuarios.
- **Riesgos de Tiempo**: Retrasos en el desarrollo. Mitigación: Planificación detallada y asignación clara de tareas.

Metodología de Desarrollo de Software

Desarrollo

En el proyecto "Tijuana Legalis" se ha seleccionado la metodología SCRUM para el desarrollo del software. Esta metodología ágil permitirá una gestión eficiente del desarrollo, priorizando la adaptabilidad y la entrega continua de valor al usuario.

Fases de SCRUM en el proyecto

1. Inicio del Proyecto: Se establecen los objetivos y se identifican los requerimientos principales. Se forma el equipo de trabajo y se definen los roles.
2. Planificación del Sprint: Se dividen las funcionalidades en incrementos manejables, organizados en el backlog del producto. Se priorizan las tareas para cada sprint.
3. Desarrollo e Implementación: Se implementan las funcionalidades establecidas en el sprint, asegurando que cada incremento del producto sea funcional.
4. Revisión del Sprint: Se evalúa el trabajo realizado, se realizan pruebas y se obtienen retroalimentaciones para mejorar el siguiente ciclo de desarrollo.
5. Retrospectiva del Sprint: Se analizan los aciertos y áreas de mejora, promoviendo la optimización del proceso en futuras iteraciones.

Ventajas de SCRUM en Tijuana Legalis:

- Permite entregas incrementales del sistema, facilitando la detección y corrección temprana de errores.
- Mejora la comunicación y la colaboración dentro del equipo de desarrollo.
- Se adapta fácilmente a cambios en los requisitos del proyecto.
- Favorece la transparencia y el seguimiento del progreso mediante reuniones diarias (Daily Stand-ups).

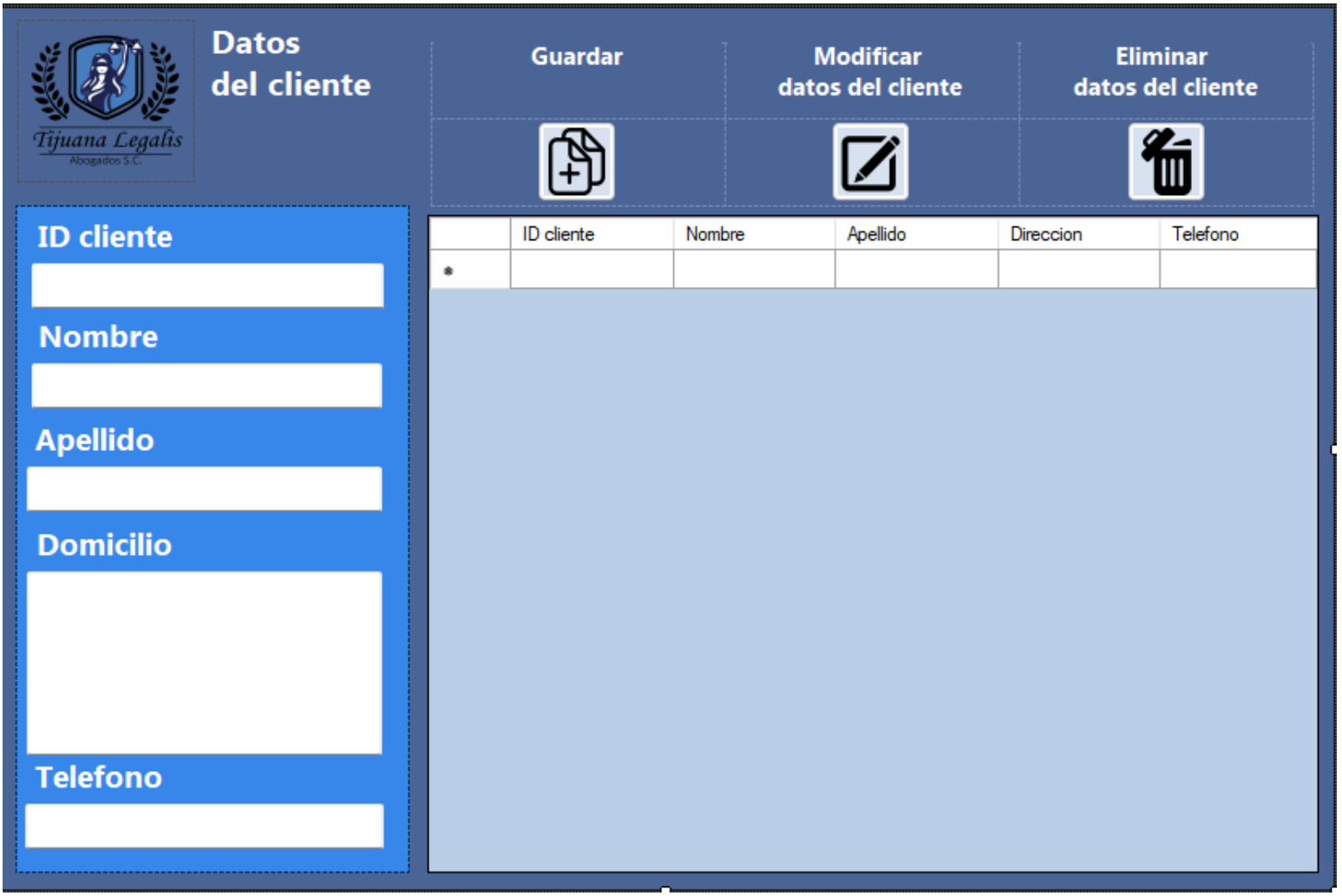
Diseño de Interfaces del Sistema

Caso de Uso: Menú de inicio

- **Interfaz:** Menú de inicio del programa
- **Descripción:** En esta ventana nos da a elegir las opciones de; -Agregar cliente -Registrar abogado -Asignar caso
- **Componentes:** Botones de, agregar cliente, registrar abogado, asignar caso.

1. Caso de Uso: Registro de Cliente

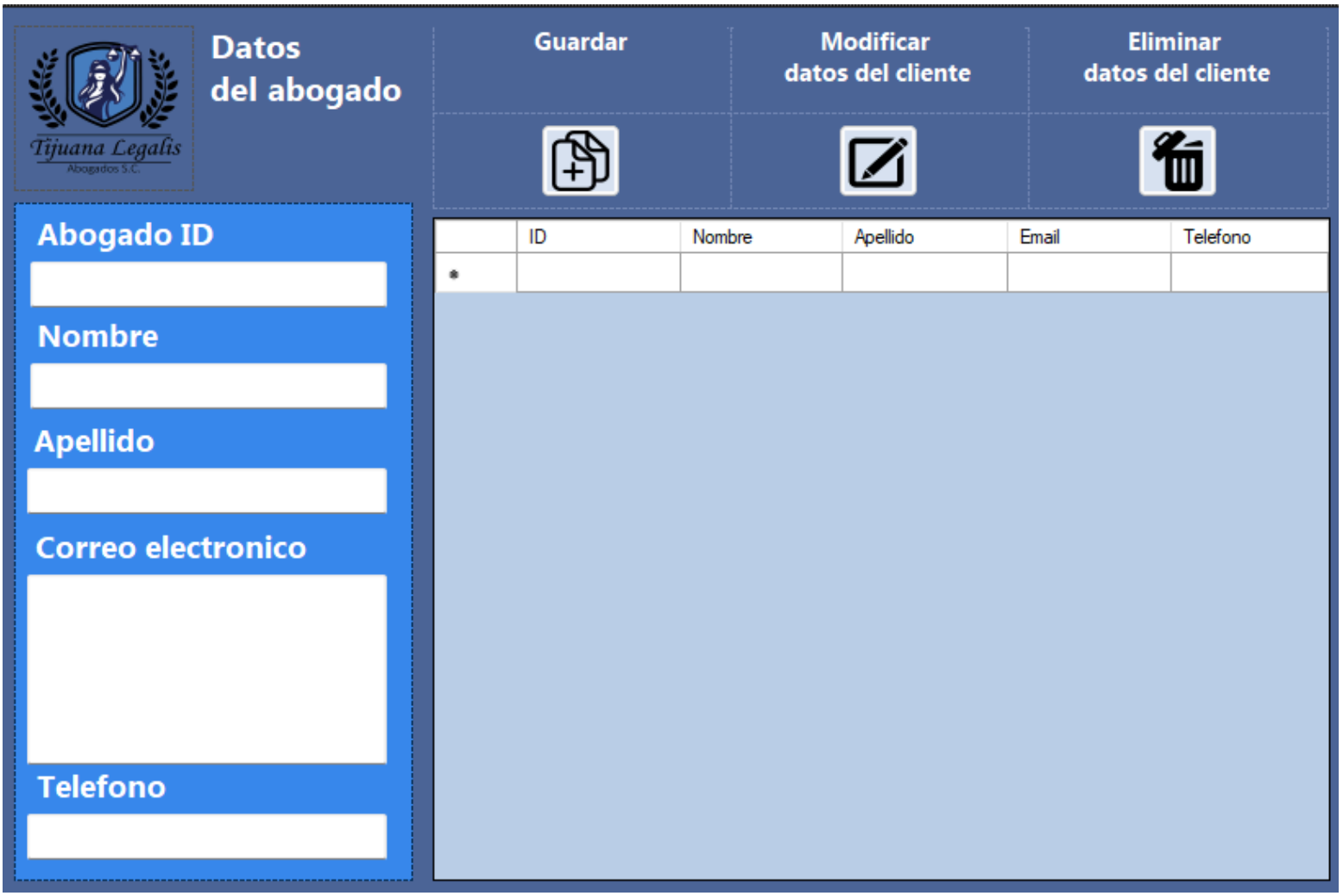
- **Interfaz:** Formulario de Registro de Cliente.



- **Descripción:** Permite ingresar la información de un nuevo cliente, incluyendo ID, nombre, dirección y contacto.
- **Componentes:** Campos de entrada, botones de guardar, modificar datos y eliminar datos.

2. Caso de Uso: Registro de Abogado

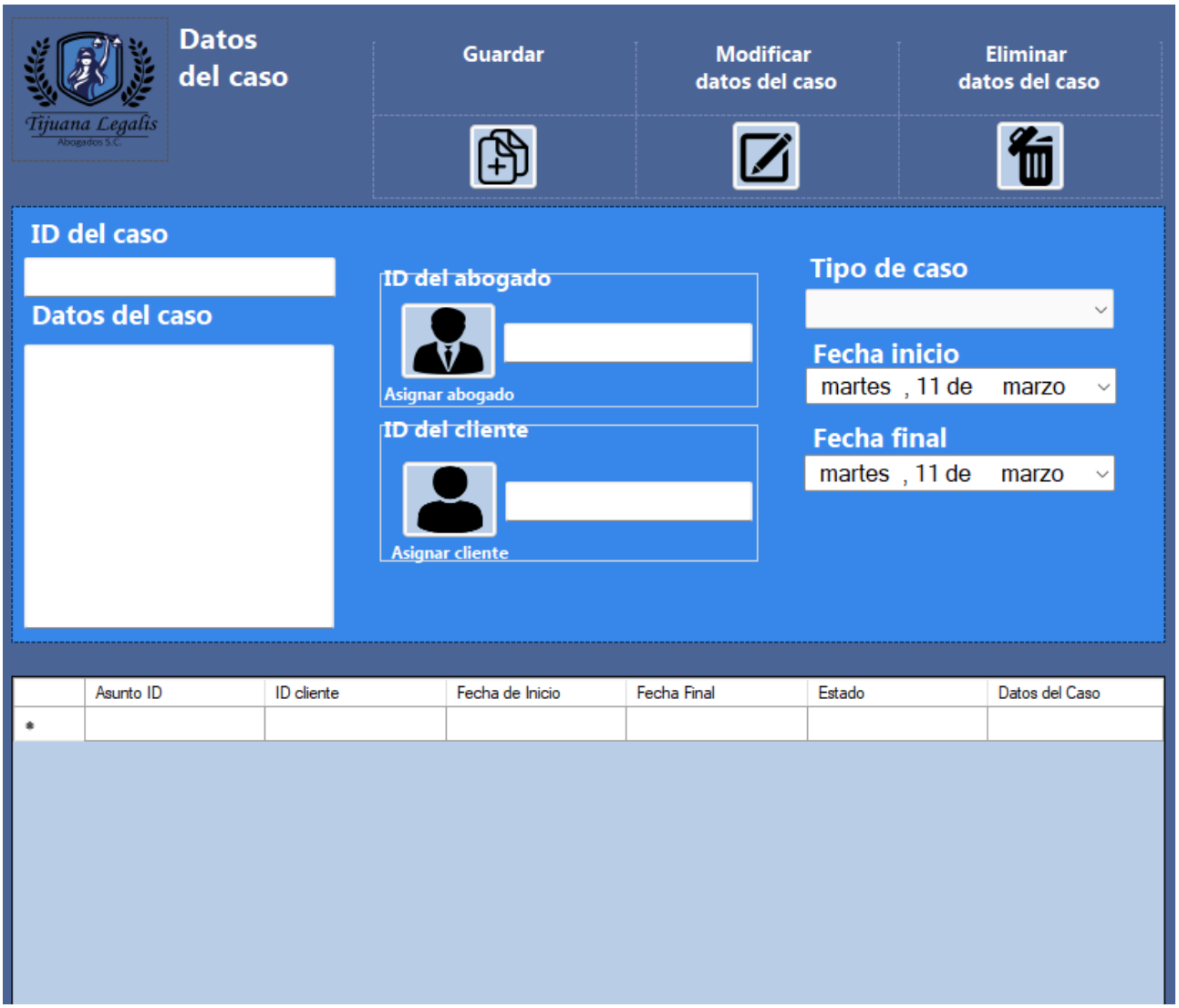
- **Interfaz:** Formulario de Registro de Abogado.



- **Descripción:** Permite registrar un abogado con su respectiva información de contacto
- **Componentes:** Campos de entrada, botones de guardar, modificar datos y eliminar datos.

3. Caso de Uso: Registro de Caso Legal

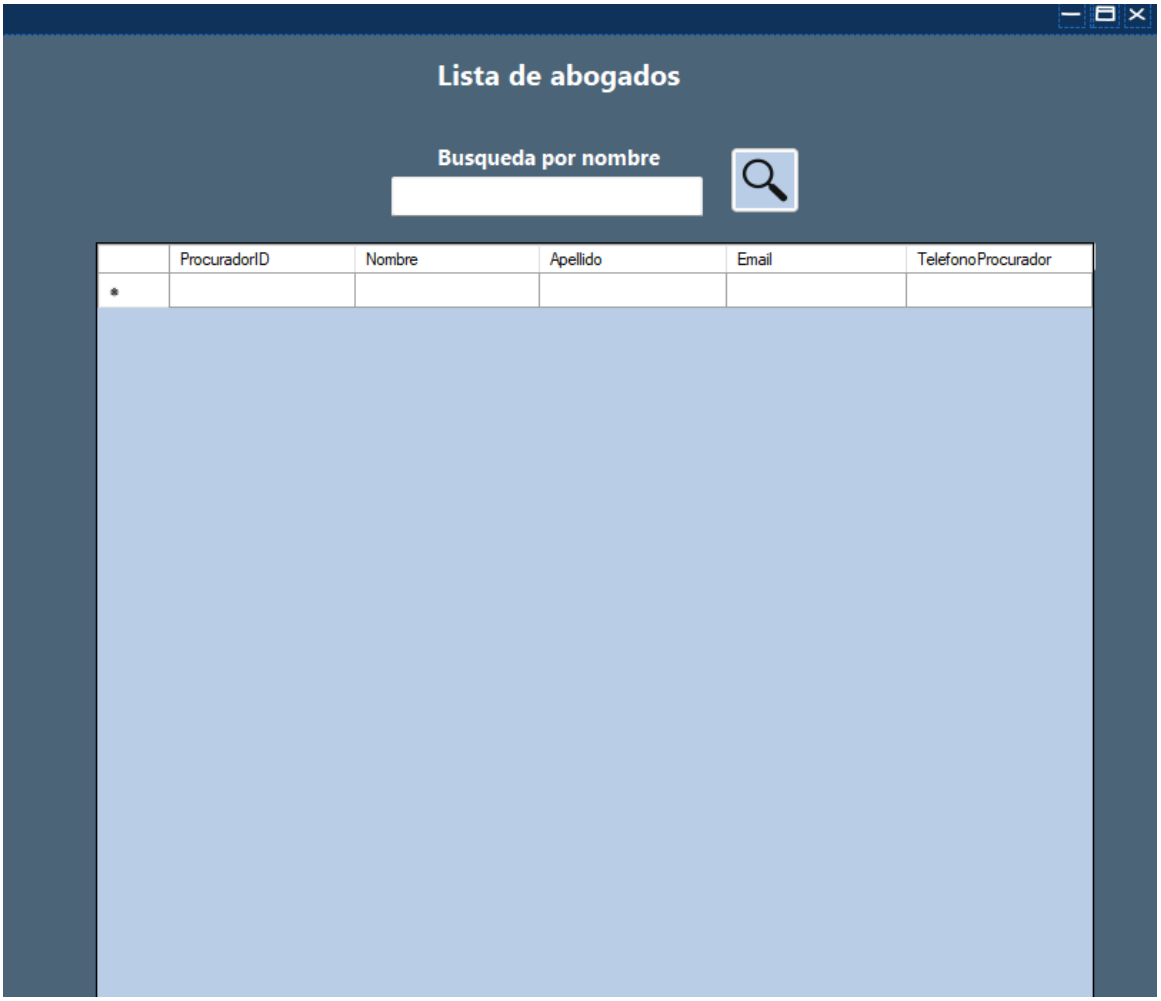
- **Interfaz:** Formulario de Caso Legal.



- **Descripción:** Permite ingresar detalles del caso, como cliente asociado, abogado, fechas y estado.
- **Componentes:** Menús desplegables, campos de texto, botones de acción.

3.1. Caso de Uso: Lista de abogados

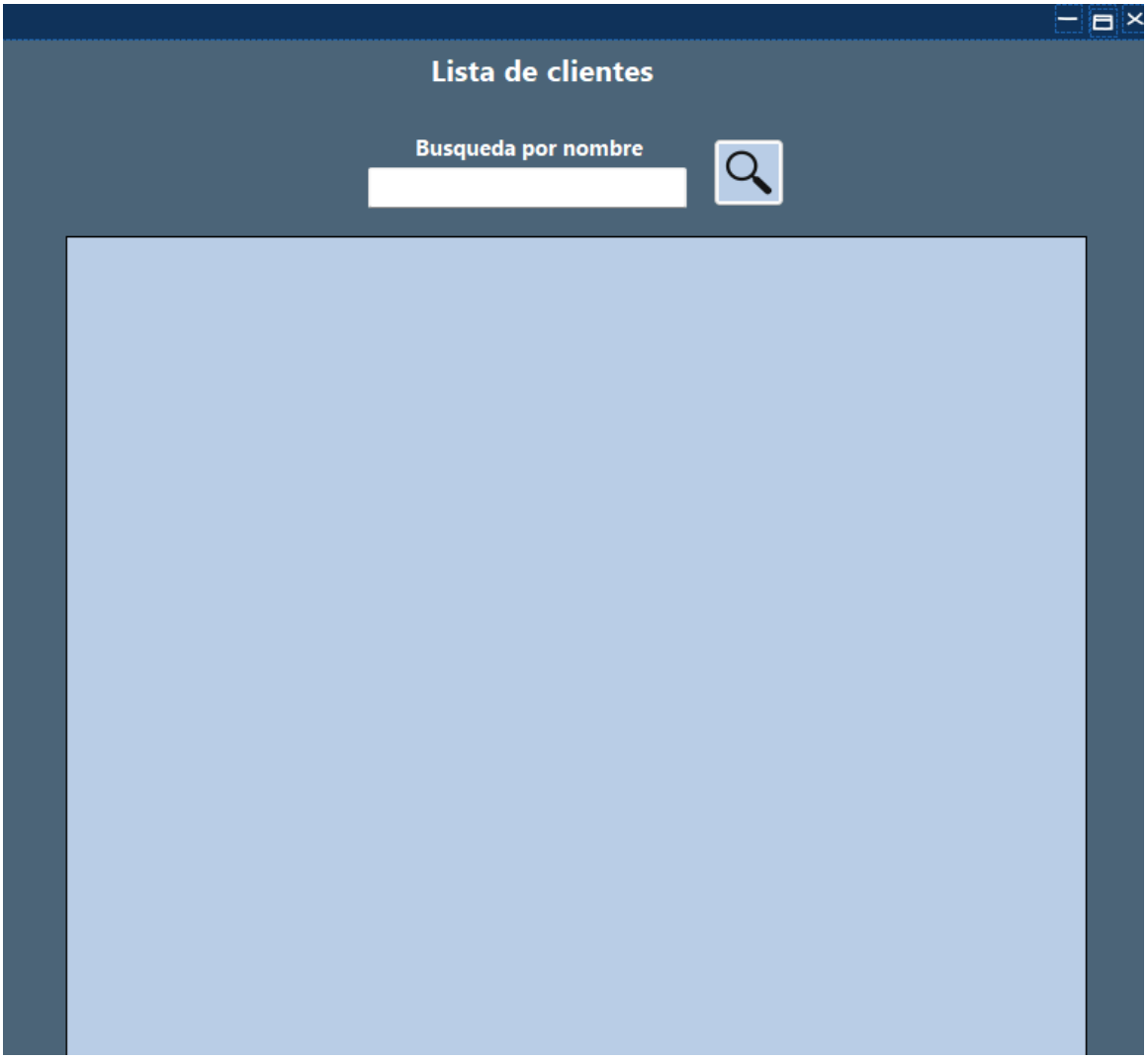
- **Interfaz:** Lista de abogados.



- **Descripción:** Puede hacerse la búsqueda de los abogados, simplemente escribiendo su nombre o buscando en la tabla.
- **Componentes:** Botón de búsqueda, tabla y textbox.

3.2. Caso de Uso: Lista de clientes

- **Interfaz:** Lista de clientes



- **Descripción:** Puede hacerse la búsqueda de los clientes, simplemente escribiendo su nombre o buscando en la tabla.
- **Componentes:** Botón de búsqueda, tabla y textbox.

Lenguaje de Programación y Sistema Gestor de Base de Datos

Selección del Lenguaje de Programación

Para el desarrollo del sistema "Tijuana Legalis" se ha seleccionado **C# con Windows Forms** como lenguaje de programación principal. C# es un lenguaje robusto, orientado a objetos y altamente compatible con la plataforma .NET, lo que facilita la integración con otros servicios de Microsoft y mejora la escalabilidad del sistema.

Razones de la elección

- **Facilidad de desarrollo:** Permite crear aplicaciones de escritorio con una interfaz gráfica intuitiva.
- **Compatibilidad:** Se integra perfectamente con Windows, que es el sistema operativo objetivo del proyecto.
- **Rendimiento:** Optimiza el uso de recursos del sistema, asegurando una ejecución eficiente.
- **Seguridad:** Proporciona un marco seguro para la gestión de datos y acceso a la base de datos.

Selección del Sistema Gestor de Base de Datos (SGBD)

El sistema utilizará **Microsoft SQL Server** como gestor de base de datos. Este SGBD es ideal para aplicaciones empresariales debido a su capacidad de manejar grandes volúmenes de datos y su integración con tecnologías de Microsoft.

Razones de la elección

- **Integración nativa con C# y .NET Framework.**
- **Alto rendimiento y escalabilidad.**
- **Seguridad avanzada con cifrado y gestión de accesos.**
- **Herramientas de administración eficientes como SQL Server Management Studio (SSMS).**

Justificación de la Compatibilidad

C# y Microsoft SQL Server forman una combinación óptima para el desarrollo del sistema, ya que ambos son productos de Microsoft y están diseñados para trabajar en conjunto. La integración entre estos permite la optimización de consultas, el manejo eficiente de datos y la implementación de funcionalidades avanzadas como procedimientos almacenados y triggers.

Codificar Procesos

Procesos Codificados y Explicación

1. **Registro de Clientes**

●**Archivo:** Form2.cs

●**Función:** Permite registrar un nuevo cliente, validando que el DNI no exista previamente y cumpla formato correcto. Los datos se envían a la base de datos.

2. **Modificación de Datos de Cliente**

●**Archivo:** Form2.cs

●**Función:** Actualiza la información de clientes existentes mediante un procedimiento almacenado.

3. **Eliminación de Cliente**

●**Archivo:** Form2.cs

●**Función:** Elimina un cliente solo si no tiene asuntos legales activos registrados.

4. **Registro de Abogado**

●**Archivo:** Form3.cs

●**Función:** Permite registrar procuradores con validaciones de campos y correo electrónico.

5. **Modificación y Eliminación de Abogado**

●**Archivo:** Form3.cs

●**Función:** Actualiza o elimina registros de procuradores según su ID.

6. **Registro de Caso Legal**

●**Archivo:** Form4.cs

●**Función:** Asocia un caso legal a un cliente y un abogado, estableciendo fechas y estado.

7. **Modificación y Eliminación de Caso Legal**

●**Archivo:** Form4.cs

●**Función:** Actualiza o elimina registros de casos con procedimientos almacenados.

8. **Visualización y Asignación desde Listados**

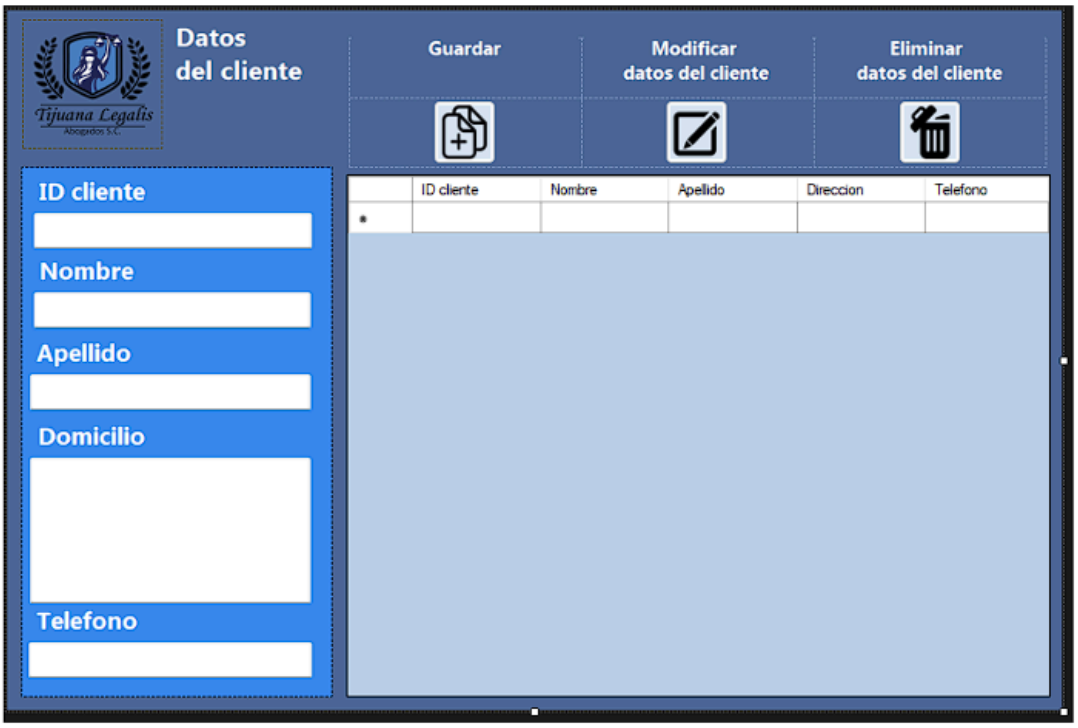
●**Archivos:** Form6.cs (abogados), Form7.cs (clientes)

●**Función:** Listan y permiten seleccionar registros para asignar a casos.

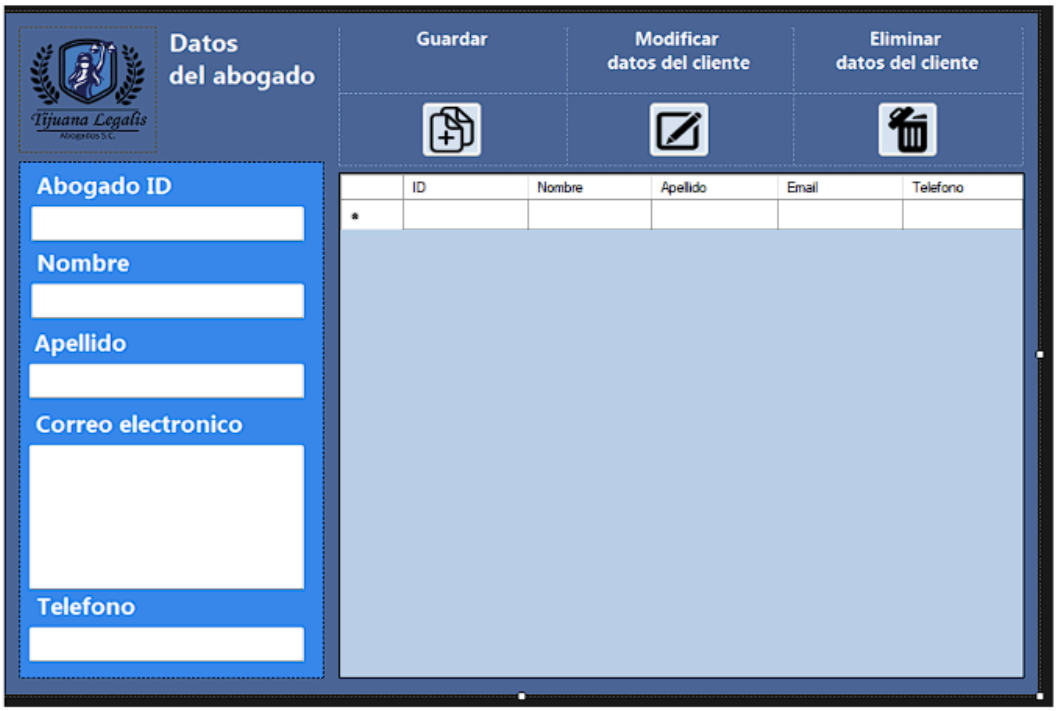
Archivos del Programa

●Inicio.cs (pantalla inicial)

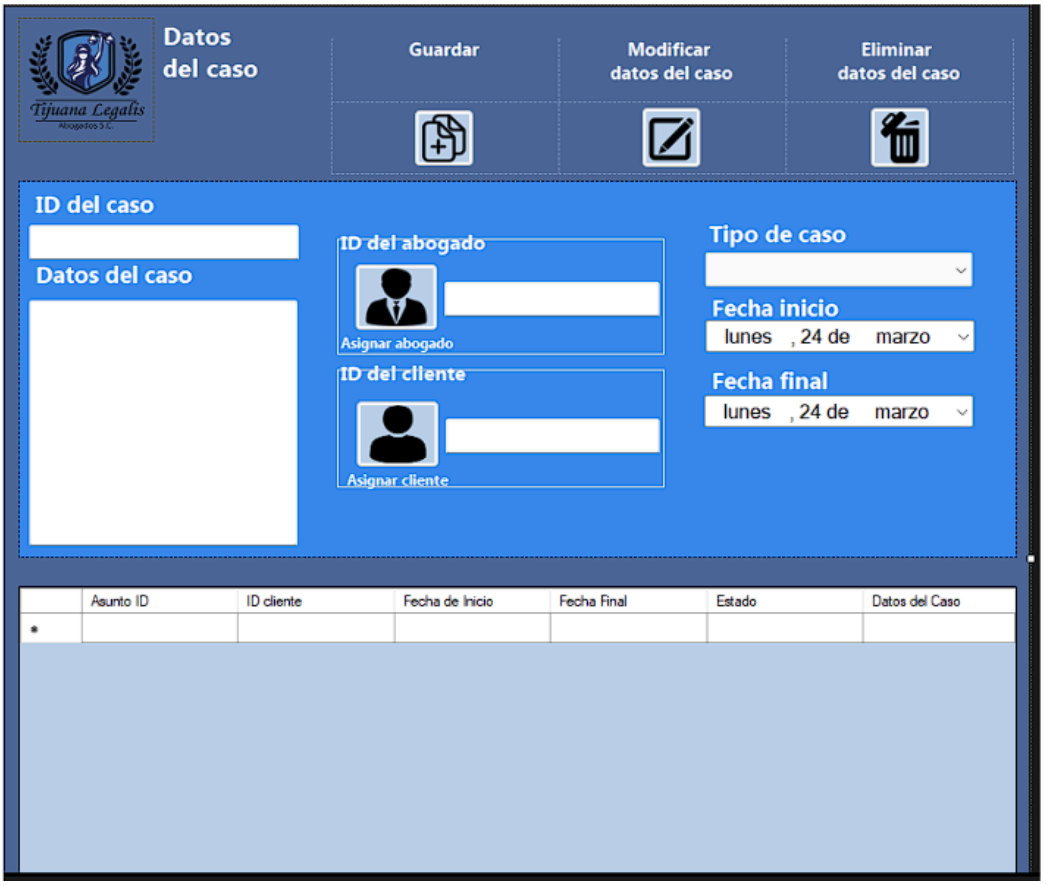
●Form2.cs (clientes)



●Form3.cs (abogados)

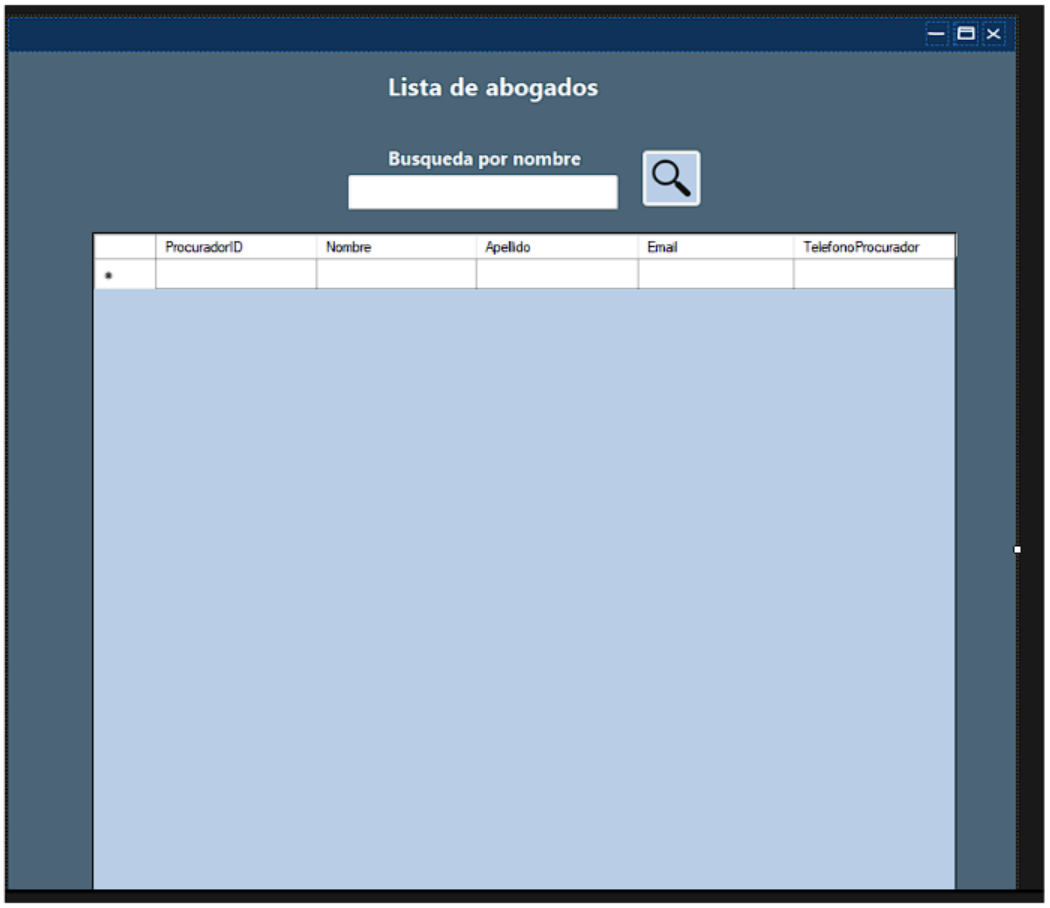


●Form4.cs (casos legales)

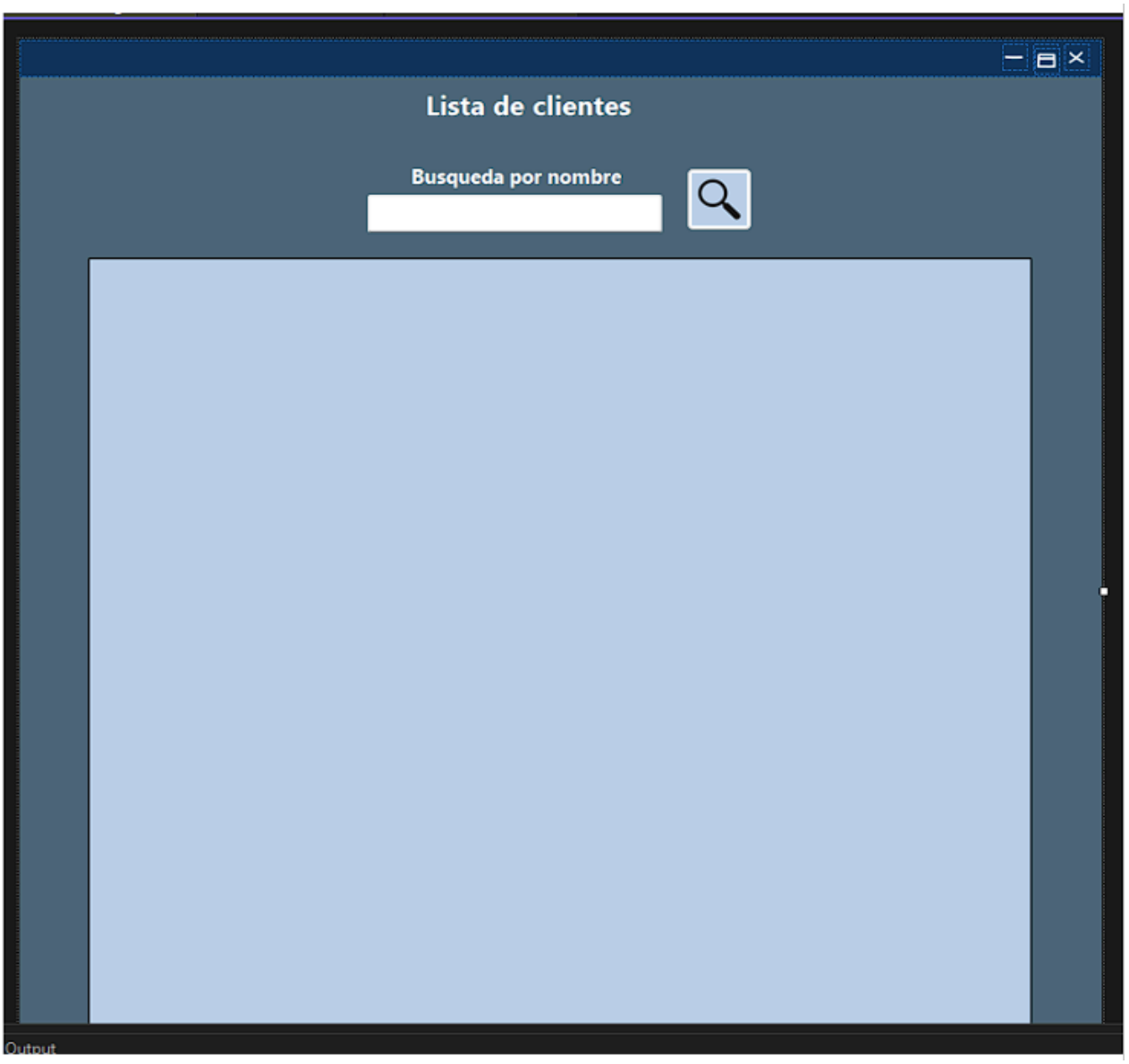


●Form5.cs (menú principal)

●Form6.cs (lista de abogados)



●Form7.cs (lista de clientes)



●Archivo SQL (creación de tablas y procedimientos almacenados).

Documentación Técnica

Arquitectura del Sistema

El sistema sigue una arquitectura **cliente-servidor**, donde la interfaz gráfica desarrollada en **C# con Windows Forms** actúa como cliente y el **Microsoft SQL Server** como servidor de base de datos.

Lenguaje y Herramientas de Desarrollo

- **Lenguaje:** C# (Windows Forms)
- **Base de datos:** Microsoft SQL Server
- **Entorno de desarrollo:** Visual Studio
- **Gestor de base de datos:** SQL Server Management Studio (SSMS)

Estructura del Código

El código fuente se organiza en clases y formularios según las entidades del sistema:

- **Inicio.cs:** Controla la autenticación y acceso al sistema.
- **Form2.cs:** Módulo de clientes (registro, modificación, eliminación).
- **Form3.cs:** Módulo de abogados (registro, modificación, eliminación).
- **Form4.cs:** Módulo de casos legales.
- **Form5.cs:** Menú principal y navegación.
- **Form6.cs y Form7.cs:** Listados de abogados y clientes.

Base de Datos

El sistema maneja las siguientes tablas clave:

- **Clientes:** Almacena información de los clientes.
- **Abogados:** Registra los procuradores del sistema.
- **Casos:** Relaciona clientes con abogados y almacena su estado.
- **Usuarios:** Maneja la autenticación de acceso.

Se han implementado procedimientos almacenados para la gestión eficiente de datos, incluyendo validaciones de integridad.

Seguridad y Mantenimiento

- Implementación de roles de usuario para restringir accesos.

- Respaldo periódico de la base de datos.
- Validaciones en la interfaz para evitar inyecciones SQL y entradas inválidas.

Anexos (código)

Visual studio Windows Forms C#

Form inicio.cs (Ventana de inicio del programa)

```
using System;
using System.Windows.Forms;

namespace Bufete_de_abogados
{
    public partial class Inicio : Form
    {
        public Inicio()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Form5 form5 = new Form5();
            form5.Show();

            this.Hide();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void button2_Click(object sender, EventArgs e)
        {

            Creditos creditos = new Creditos();
            creditos.Show();
        }
```

```
        private void label1_Click(object sender, EventArgs e)
        {
            Form5 form5 = new Form5();
            form5.Show();

            this.Hide();
        }

        private void button2_Click_1(object sender, EventArgs e)
        {
            Application.Exit();
        }
    }
}
```

Form 5.cs (Ventana del menú de inicio)

```
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;

namespace Bufete_de_abogados
{
    public partial class Form5 : Form
    {
        public Form5()
        {
            InitializeComponent();
        }
```

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

        private void btn_maximizar_Click(object sender, EventArgs e)
        {
            this.WindowState = FormWindowState.Maximized;
            btn_maximizar.Visible = false;
            btn_restaurar.Visible = true;
        }

        private void btn_restaurar_Click(object sender, EventArgs e)
        {
            this.WindowState = FormWindowState.Normal;
            btn_restaurar.Visible = false;
            btn_maximizar.Visible = true;
        }

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



        //con  esto se mueve  la ventana
        [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 IParam);

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

```
        // Variable global para almacenar el formulario actual abierto
        private Form formularioActual = null;

        // Método para abrir o cerrar el formulario en el panel
        private void AbrirOcerrarFormularioEnPanel(Form formulario)
        {
            // Si el formulario que se quiere abrir ya está abierto, se cierra
            if (formularioActual != null && formularioActual.GetType() ==
formulario.GetType())
            {
                formularioActual.Close();
                formularioActual = null;
                return;
            }

            // Si hay un formulario ya abierto, se cierra
            if (this.panel_contenedor.Controls.Count > 0)
                this.panel_contenedor.Controls.RemoveAt(0);

            // Configurar el nuevo formulario y mostrarlo
            formulario.TopLevel = false;
            formulario.Dock = DockStyle.Fill;
            this.panel_contenedor.Controls.Add(formulario);
            this.panel_contenedor.Tag = formulario;
            formulario.Show();

            // Guardar la referencia al formulario actual
            formularioActual = formulario;
        }

        private void btnCrearexpediente_Click(object sender, EventArgs e)
        {
            AbrirOcerrarFormularioEnPanel(new Form2());
        }

        private void btnDatosdelabogado_Click(object sender, EventArgs e)
        {
            AbrirOcerrarFormularioEnPanel(new Form3());
        }
```

```
        private void btnAsignaciondecasos_Click(object sender, EventArgs e)
        {
            AbrirOcerrarFormularioEnPanel(new Form4());
        }

        private void btnFormularioActual_Click(object sender, EventArgs e)
        {
            // Abrir o cerrar el Form5 (tu formulario actual)
            AbrirOcerrarFormularioEnPanel(new Form5());
        }

        private void panel_contenedor_Paint(object sender, PaintEventArgs e)
        {

        }
    }
}
```

Form 2.cs (Ventana para ingresar los datos del cliente)

```
using System;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Bufete_de_abogados
{
    public partial class Form2 : Form
    {
        DataClasses1DataContext DB = new DataClasses1DataContext();
        public Form2()
        {
            InitializeComponent();
            actualizar();
            toolTip1.SetToolTip(btn_Guardar, "Guardar");
            toolTip2.SetToolTip(btnModificarexpediente, "Modificar Expediente");
            toolTip3.SetToolTip(btnBorrarexpediente, "Borrar Expediente");
            panel1.BackColor = Color.FromArgb(166, 57, 136, 236);
            Estilo();
        }

        private void btn_Guardar_Click(object sender, EventArgs e)
        {
            // Validar que los campos no estén vacíos
            if (string.IsNullOrWhiteSpace(txtDNI.Text))
            {
                MessageBox.Show("El campo DNI no puede estar vacío.");
                return;
            }
            if (txtDNI.Text.Length != 10)
            {
                MessageBox.Show("El campo DNI debe tener exactamente 10 caracteres.");
                return;
            }
```

```
            // Verificar si el DNI ya existe en la base de datos
            var existeDNI = DB.Clientes.Any(c => c.DNI_clientes == txtDNI.Text);
            if (existeDNI)
            {
                MessageBox.Show("El DNI ingresado ya existe en la base de datos. Por favor,
ingrese un DNI único.", "DNI Duplicado", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            if (string.IsNullOrWhiteSpace(txtNombre.Text))
            {
                MessageBox.Show("El campo Nombre no puede estar vacío.");
                return;
            }
            if (string.IsNullOrWhiteSpace(txtApellido.Text))
            {
                MessageBox.Show("El campo Apellido no puede estar vacío.");
                return;
            }
            if (string.IsNullOrWhiteSpace(txtDomicilio.Text))
            {
                MessageBox.Show("El campo Domicilio no puede estar vacío.");
                return;
            }
            if (string.IsNullOrWhiteSpace(txtTelefono.Text) || !txtTelefono.Text.All(char.IsDigit))
            {
                MessageBox.Show("El campo Teléfono debe contener solo números.");
                return;
            }
            if (txtTelefono.Text.Length != 10)
            {
                MessageBox.Show("El campo Teléfono debe tener exactamente 10 dígitos.");
                return;
            }
            // Nueva expresión regular que acepta letras, espacios y caracteres especiales
            if (!System.Text.RegularExpressions.Regex.IsMatch(txtNombre.Text,
@"^[a-zA-ZáéíóúÁÉÍÓÚñÑ\s]+$"))
            {
                MessageBox.Show("El campo de nombre solo acepta letras y espacios");
```

```
            return;
        }
        if (!System.Text.RegularExpressions.Regex.IsMatch(txtApellido.Text,
@"^[a-zA-ZáéíóúÁÉÍÓÚñÑ\s]+$"))
        {
            MessageBox.Show("El campo de apellido solo acepta letras y espacios");
            return;
        }

        try
        {
            DB.ingresar_clientes(txtDNI.Text, txtNombre.Text, txtApellido.Text,
txtDomicilio.Text, txtTelefono.Text);
            MessageBox.Show("Cliente guardado exitosamente.", "Información",
MessageBoxButtons.OK, MessageBoxIcon.Information);

            // Limpia los campos después de guardar
            txtDNI.Clear();
            txtApellido.Clear();
            txtNombre.Clear();
            txtDomicilio.Clear();
            txtTelefono.Clear();

            // Actualiza la vista
            actualizar();
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Error al guardar el cliente: {ex.Message}", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }


    private void btn_Consultar_Click(object sender, EventArgs e)
    {
        DB.LEER_ingresar_clientes();

    }
```

```
        private void btnModificarexpediente_Click(object sender, EventArgs e)
        {
            // Asegúrate de que txtDNI tenga el DNI del cliente que deseas actualizar
            //DB.ActualizarCliente();

            // Validar los campos como se hizo anteriormente
            if (string.IsNullOrWhiteSpace(txtDNI.Text))
            {
                MessageBox.Show("El campo DNI no puede estar vacío.");
                return;
            }

            if (string.IsNullOrWhiteSpace(txtNombre.Text))
            {
                MessageBox.Show("El campo Nombre no puede estar vacío.");
                return;
            }

            if (string.IsNullOrWhiteSpace(txtApellido.Text))
            {
                MessageBox.Show("El campo Apellido no puede estar vacío.");
                return;
            }

            if (string.IsNullOrWhiteSpace(txtDomicilio.Text))
            {
                MessageBox.Show("El campo Domicilio no puede estar vacío.");
                return;
            }

            if (string.IsNullOrWhiteSpace(txtTelefono.Text) || !txtTelefono.Text.All(char.IsDigit))
            {
                MessageBox.Show("El campo Teléfono debe contener solo números y no
puede estar vacío.");
                return;
            }

            DB.ActualizarCliente(txtDNI.Text,txtNombre.Text, txtApellido.Text,
```

```
txtDomicilio.Text, txtTelefono.Text);
            actualizar();
        }

        private void btnBorrarexpediente_Click(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtDNI.Text))
            {
                MessageBox.Show("Por favor, ingrese un DNI válido.", "Advertencia",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }

            // Verificar si el cliente tiene asuntos activos
            var asuntoEnCurso = DB.Asuntos.Any(a => a.DNI_clientes == txtDNI.Text &&
a.Estado != "Cerrado");

            if (asuntoEnCurso)
            {
                MessageBox.Show("No se puede borrar el cliente porque tiene un asunto en
curso.",
                                "Advertencia", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }
            DB.BorrarClienteConVerificacion(txtDNI.Text);
            actualizar();
            MessageBox.Show("Cliente eliminado correctamente.", "Información",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        }

        private void Form2_Load(object sender, EventArgs e)
        {
            // TODO: esta línea de código carga datos en la tabla
'abogados3DataSet.Clientes' Puede moverla o quitarla según sea necesario.
            this.clientesTableAdapter2.Fill(this.abogados3DataSet.Clientes);
            // TODO: esta línea de código carga datos en la tabla
'nUEVO_GABINETE_DE_ABOGADOSDataSet21.Clientes' Puede moverla o quitarla
según sea necesario.
```

```
this.clientesTableAdapter1.Fill(this.nUEVO_GABINETE_DE_ABOGADOSDataSet21.Clientes);
            // TODO: esta línea de código carga datos en la tabla 'nUEVO_GABINETE_DE_ABOGADOSDataSet18.Clientes' Puede moverla o quitarla según sea necesario.

this.clientesTableAdapter.Fill(this.nUEVO_GABINETE_DE_ABOGADOSDataSet18.Clientes);
            // TODO: esta línea de código carga datos en la tabla 'nUEVO_GABINETE_DE_ABOGADOSDataSet.Clientes' Puede moverla o quitarla según sea necesario.

//this.clientesTableAdapter.Fill(this.nUEVO_GABINETE_DE_ABOGADOSDataSet.Clientes);

        }

        void actualizar()
        {
            this.clientesTableAdapter2.Fill(this.abogados3DataSet.Clientes);

        }

        private void Estilo()
        {
            dataGridView1.ColumnHeadersDefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter; // Centrar encabezados
            dataGridView1.ColumnHeadersDefaultCellStyle.Font = new Font("Segoe UI", 10, FontStyle.Bold); // Negritas en encabezados
            dataGridView1.ColumnHeadersDefaultCellStyle.ForeColor = ColorTranslator.FromHtml("#212121"); // Color de texto de encabezados
            dataGridView1.EnableHeadersVisualStyles = false; // Desactivar estilos visuales para aplicar el estilo personalizado
            dataGridView1.ColumnHeadersDefaultCellStyle.BackColor = ColorTranslator.FromHtml("#F5F5F5"); // Color de fondo de encabezados

            dataGridView1.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
```

```
        }

        private void button1_Click_1(object sender, EventArgs e)
        {
            Close();
            Form1 regresraform1 = new Form1();
            regresraform1.Show();
        }

        private void dataGridView1_RowHeaderMouseClick(object sender,
DataGridViewCellMouseEventArgs e)
        {
            if (e.RowIndex >= 0)
            {
                DataGridViewRow selectedRow = dataGridView1.Rows[e.RowIndex];
                // Aquí puedes acceder a la fila seleccionada y realizar acciones en función de
la fila.
                // Por ejemplo, puedes obtener el valor de una celda de la fila:
selectedRow.Cells["NombreDeLaColumna"].Value

                txtDNI.Text = selectedRow.Cells[0].Value.ToString();
                txtNombre.Text = selectedRow.Cells[1].Value.ToString();
                txtApellido.Text = selectedRow.Cells[2].Value.ToString();
                txtDomicilio.Text = selectedRow.Cells[3].Value.ToString();
                txtTelefono.Text = selectedRow.Cells[4].Value.ToString();
            }
        }



        private void txtTelefono_KeyPress(object sender, KeyPressEventArgs e)
        {
            txtTelefono.MaxLength = 10;
            // Verifica si la tecla presionada es un número o la tecla de retroceso (Backspace)
            if (!char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar))
            {
                e.Handled = true;  // Cancela la entrada si no es un número
            }
        }
```

```
        private void txtDNI_KeyPress(object sender, KeyPressEventArgs e)
        {
            txtDNI.MaxLength = 10;

            // Verifica si la tecla presionada es un número o la tecla de retroceso (Backspace)
            if (!char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar))
            {
                e.Handled = true;  // Cancela la entrada si no es un número
            }
        }

        private void txtNombre_TextChanged(object sender, EventArgs e)
        {

        }

        private void txtDomicilio_TextChanged(object sender, EventArgs e)
        {

        }

        private void txtNombre_KeyPress(object sender, KeyPressEventArgs e)
        {
            // Verifica si el carácter no es letra, espacio o una tecla de control (como
retroceso)
            if (!char.IsLetter(e.KeyChar) && !char.IsControl(e.KeyChar) &&
!char.IsWhiteSpace(e.KeyChar))
            {
                // Cancela el evento, es decir, no permite que se escriba el carácter
                e.Handled = true;
            }
        }

        private void txtApellido_KeyPress(object sender, KeyPressEventArgs e)
        {
            // Verifica si el carácter no es letra, espacio o una tecla de control (como
retroceso)
            if (!char.IsLetter(e.KeyChar) && !char.IsControl(e.KeyChar) &&
```

```
!char.IsWhiteSpace(e.KeyChar))
            {
                // Cancela el evento, es decir, no permite que se escriba el carácter
                e.Handled = true;
            }
        }
    }
}
```

Form 3.cs (Ventana para ingresar los datos del abogado)

```
using System;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;

namespace Bufete_de_abogados
{
    public partial class Form3 : Form
    {
        DataClasses1DataContext DB = new DataClasses1DataContext();
        public Form3()
        {
            InitializeComponent();
            actualizar();
            toolTip1.SetToolTip(btn_Guardar, "Guardar");
            toolTip2.SetToolTip(btnModificarexpediente, "Modificar Expediente");
            toolTip3.SetToolTip(btnBorrarexpediente, "Borrar datos del procurador");
            panel1.BackColor = Color.FromArgb(166, 57, 136, 236);
            Estilo();
        }

        private void Form3_Load(object sender, EventArgs e)
        {
            // TODO: esta línea de código carga datos en la tabla
'abogados3DataSet2.Procuradores' Puede moverla o quitarla según sea necesario.
```

```
            this.procuradoresTableAdapter1.Fill(this.abogados3DataSet2.Procuradores);
            // TODO: esta línea de código carga datos en la tabla
'nUEVO_GABINETE_DE_ABOGADOSDataSet1.Procuradores' Puede moverla o quitarla
según sea necesario.
            //
this.procuradoresTableAdapter.Fill(this.nUEVO_GABINETE_DE_ABOGADOSDataSet1.Pro
curadores);

        }

        void actualizar()
        {
            this.procuradoresTableAdapter1.Fill(this.abogados3DataSet2.Procuradores);

        }

        private void dataGridView1_RowHeaderMouseClick(object sender,
DataGridViewCellMouseEventArgs e)
        {
            if (e.RowIndex >= 0)
            {
                DataGridViewRow selectedRow = dataGridView1.Rows[e.RowIndex];
                // Aquí puedes acceder a la fila seleccionada y realizar acciones en función de
la fila.
                // Por ejemplo, puedes obtener el valor de una celda de la fila:
selectedRow.Cells["NombreDeLaColumna"].Value
                txtIDprocurador.Text = selectedRow.Cells[0].Value.ToString();
                txtNombre.Text = selectedRow.Cells[1].Value.ToString();
                txtApellido.Text = selectedRow.Cells[2].Value.ToString();
                txtCorreo.Text = selectedRow.Cells[3].Value.ToString();
                txtTelefono.Text = selectedRow.Cells[4].Value.ToString();


            }
        }

        private void Estilo()
        {
            dataGridView1.ColumnHeadersDefaultCellStyle.Alignment =
```

```
DataGridViewContentAlignment.MiddleCenter; // Centrar encabezados
            dataGridView1.ColumnHeadersDefaultCellStyle.Font = new Font("Segoe UI", 10,
FontStyle.Bold); // Negritas en encabezados
            dataGridView1.ColumnHeadersDefaultCellStyle.ForeColor =
ColorTranslator.FromHtml("#212121"); // Color de texto de encabezados
            dataGridView1.EnableHeadersVisualStyles = false; // Desactivar estilos visuales
para aplicar el estilo personalizado
            dataGridView1.ColumnHeadersDefaultCellStyle.BackColor =
ColorTranslator.FromHtml("#F5F5F5"); // Color de fondo de encabezados

            dataGridView1.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
        }


        private void txtIDprocurador_KeyPress(object sender, KeyPressEventArgs e)
        {
            txtIDprocurador.MaxLength = 10;
            // Verifica si la tecla presionada es un número o la tecla de retroceso (Backspace)
            if (!char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar))
            {
                e.Handled = true;  // Cancela la entrada si no es un número
            }
        }

        private void txtTelefono_KeyPress(object sender, KeyPressEventArgs e)
        {
            txtTelefono.MaxLength = 10;
            // Verifica si la tecla presionada es un número o la tecla de retroceso (Backspace)
            if (!char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar))
            {
                e.Handled = true;  // Cancela la entrada si no es un número
            }
        }

        private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
```

```
        }

        private void btn_Guardar_Click(object sender, EventArgs e)
        {
            // Validar que los campos no estén vacíos
            if (string.IsNullOrWhiteSpace(txtIDprocurador.Text))
            {
                MessageBox.Show("El campo ID no puede estar vacío.");
                return;
            }
            if (txtIDprocurador.Text.Length != 10)
            {
                MessageBox.Show("El campo ID debe tener exactamente 10 caracteres.");
                return;
            }

            // Verificar que el ID no esté repetido
            var existeID = DB.Procuradores.Any(p => p.ProcuradorID ==
txtIDprocurador.Text);
            if (existeID)
            {
                MessageBox.Show("El ID ingresado ya existe en la base de datos. Por favor,
ingrese un ID único.", "ID Duplicado", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }


            if (string.IsNullOrWhiteSpace(txtTelefono.Text) || !txtTelefono.Text.All(char.IsDigit))
            {
                MessageBox.Show("El campo Teléfono debe contener solo números.");
                return;
            }
            if (txtTelefono.Text.Length != 10)
            {
                MessageBox.Show("El campo Teléfono debe tener exactamente 10 dígitos.");
                return;
            }
            // Nueva expresión regular que acepta letras, espacios y caracteres especiales
            if (!System.Text.RegularExpressions.Regex.IsMatch(txtNombre.Text,
```

```
@"^[a-zA-ZáéíóúÁÉÍÓÚñÑ\s]+$"))
            {
                MessageBox.Show("El campo de nombre solo acepta letras y espacios");
                return;
            }
            if (!System.Text.RegularExpressions.Regex.IsMatch(txtApellido.Text,
@"^[a-zA-ZáéíóúÁÉÍÓÚñÑ\s]+$"))
            {
                MessageBox.Show("El campo de apellido solo acepta letras y espacios");
                return;
            }
            if (!txtCorreo.Text.Contains("@"))
            {
                MessageBox.Show("El correo debe contener un '@'.");
                return;
            }

            ////
            try
            {
                DB.ingresar_Procuradores(txtIDprocurador.Text, txtNombre.Text,
txtApellido.Text, txtCorreo.Text, txtTelefono.Text);
                MessageBox.Show("Abogadoo guardado exitosamente.", "Información",
MessageBoxButtons.OK, MessageBoxIcon.Information);

                // Limpia los campos después de guardar
                txtIDprocurador.Clear();
                txtApellido.Clear();
                txtNombre.Clear();
                txtCorreo.Clear();
                txtTelefono.Clear();

                // Actualiza la vista
                actualizar();
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Error al guardar el cliente: {ex.Message}", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
            }

            /////
            //DB.ingresar_Procuradores(txtIDprocurador.Text, txtNombre.Text, txtApellido.Text,
txtCorreo.Text, txtTelefono.Text);
            actualizar();

        }

        private void btnModificarexpediente_Click(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtNombre.Text))
            {
                MessageBox.Show("El campo nombre no puede estar vacío.");
                return;
            }

            if (string.IsNullOrWhiteSpace(txtApellido.Text))
            {
                MessageBox.Show("El campo apellido no puede estar vacío.");
                return;
            }

            if (string.IsNullOrWhiteSpace(txtCorreo.Text))
            {
                MessageBox.Show("El campo correo no puede estar vacío.");
                return;
            }

            if (string.IsNullOrWhiteSpace(txtTelefono.Text))
            {
                MessageBox.Show("El campo telefono no puede estar vacío.");
                return;
            }

            DB.ActualizarProcurador(txtIDprocurador.Text, txtNombre.Text, txtApellido.Text,
txtCorreo.Text, txtTelefono.Text);
            actualizar();
        }
```

```
        private void btnBorrarexpediente_Click(object sender, EventArgs e)
        {
            DB.Borrar_Procuradores(txtIDprocurador.Text);
            actualizar();
        }

        private void txtApellido_TextChanged(object sender, EventArgs e)
        {

        }

        private void txtNombre_KeyPress(object sender, KeyPressEventArgs e)
        {
            // Verifica si el carácter no es letra, espacio o una tecla de control (como
retroceso)
            if (!char.IsLetter(e.KeyChar) && !char.IsControl(e.KeyChar) &&
!char.IsWhiteSpace(e.KeyChar))
            {
                // Cancela el evento, es decir, no permite que se escriba el carácter
                e.Handled = true;
            }
        }

        private void txtApellido_KeyPress(object sender, KeyPressEventArgs e)
        {
            // Verifica si el carácter no es letra, espacio o una tecla de control (como
retroceso)
            if (!char.IsLetter(e.KeyChar) && !char.IsControl(e.KeyChar) &&
!char.IsWhiteSpace(e.KeyChar))
            {
                // Cancela el evento, es decir, no permite que se escriba el carácter
                e.Handled = true;
            }
        }
    }
}
```

Form 4.cs (Ventana para ingresar los datos del caso)

```
using System;
using System.Drawing;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Bufete_de_abogados
{
    public partial class Form4 : Form
    {
        DataClasses1DataContext DB = new DataClasses1DataContext();
        public Form4()
        {
            InitializeComponent();
            //actualizar();
            toolTip1.SetToolTip(btnGuardar, "Guardar");
            toolTip2.SetToolTip(btnModificardatosdelcaso, "Modificar Datos del Caso");
            toolTip3.SetToolTip(btnborrarexpediente, "Borrar Expediente");
            panel1.BackColor = Color.FromArgb(166, 57, 136, 236);

            // Configurar el ComboBox
            comboBoxEstado.Items.Add("Divorcio");
            comboBoxEstado.Items.Add("Pension alimenticia");
            comboBoxEstado.Items.Add("Herencia");
            comboBoxEstado.Items.Add("Despido injustificado");
            comboBoxEstado.Items.Add("Invasion de propiedad");
            Estilo();
        }

        public void SetProcuradorID(string procuradorID)
        {
            ProcuradorIDtxt.Text = procuradorID;
        }

        public void SetClienteDNI(string ClienteID)
        {
```

```
            ClienteIDtxt.Text = ClienteID;
        }

        private void btnGuardar_Click(object sender, EventArgs e)
        {
            // Validar que los campos no estén vacíos
            if (string.IsNullOrWhiteSpace(txtDNI.Text))
            {
                MessageBox.Show("El campo ID no puede estar vacío.");
                return;
            }
            if (txtDNI.Text.Length != 10)
            {
                MessageBox.Show("El campo ID debe tener exactamente 10 caracteres.");
                return;
            }
            if (comboBoxEstado.SelectedItem == null)
            {
                MessageBox.Show("Por favor, seleccione un estado.");
                return;
            }

            // Validar que la fecha de inicio no sea posterior a la fecha final
            if (dateTimePicker2.Value > dateTimePicker1.Value)
            {
                MessageBox.Show("La fecha de inicio no puede ser posterior a la fecha final.");
                return;
            }

            try
            {
                // Llamar al procedimiento para ingresar asuntos
                DB.ingresar_Asuntos(txtDNI.Text, ClienteIDtxt.Text, dateTimePicker1.Value,
dateTimePicker2.Value, comboBoxEstado.SelectedItem.ToString(), txtDatodelcaso.Text);
                MessageBox.Show("Asunto guardado exitosamente.");
            }
            catch (Exception ex)
            {
                // Manejo de excepciones
```

```
                MessageBox.Show("Ocurrió un error al guardar el asunto: " + ex.Message);
            }

            // Actualizar la vista después de guardar
            actualizar();
        }


        private void btnModificardatosdelcaso_Click(object sender, EventArgs e)
        {

            DB.ActualizarAsunto(txtDNI.Text,dateTimePicker2.Value, dateTimePicker1.Value,
comboBoxEstado.SelectedItem.ToString(), txtDatodelcaso.Text);
            actualizar();
        }

        private void btnborrarexpediente_Click(object sender, EventArgs e)
        {
            DB.Borrar_Asunto(txtDNI.Text);
            actualizar();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Close();
            Form1 regresraform1 = new Form1();
            regresraform1.Show();
        }

        private void Form4_Load(object sender, EventArgs e)
        {
            // TODO: esta línea de código carga datos en la tabla
'abogados3DataSet1.Asuntos' Puede moverla o quitarla según sea necesario.
            this.asuntosTableAdapter3.Fill(this.abogados3DataSet1.Asuntos);
            // TODO: esta línea de código carga datos en la tabla
'nUEVO_GABINETE_DE_ABOGADOSDataSet19.Asuntos' Puede moverla o quitarla
según sea necesario.

this.asuntosTableAdapter2.Fill(this.nUEVO_GABINETE_DE_ABOGADOSDataSet19.Asunt
```

```
os);

            // TODO: esta línea de código carga datos en la tabla
'nUEVO_GABINETE_DE_ABOGADOSDataSet16.Asuntos' Puede moverla o quitarla
según sea necesario.

//this.asuntosTableAdapter1.Fill(this.nUEVO_GABINETE_DE_ABOGADOSDataSet16.Asu
ntos);
            // TODO: esta línea de código carga datos en la tabla
'nUEVO_GABINETE_DE_ABOGADOSDataSet13.Asuntos' Puede moverla o quitarla
según sea necesario.
            //
this.asuntosTableAdapter.Fill(this.nUEVO_GABINETE_DE_ABOGADOSDataSet13.Asunto
s);
            // TODO: esta línea de código carga datos en la tabla
'nUEVO_GABINETE_DE_ABOGADOSDataSet12.Asuntos' Puede moverla o quitarla
según sea necesario.

//this.asuntosTableAdapter.Fill(this.nUEVO_GABINETE_DE_ABOGADOSDataSet12.Asunt
os);
        }
        void actualizar()
        {


//this.asuntosTableAdapter.Fill(this.nUEVO_GABINETE_DE_ABOGADOSDataSet13.Asunt
os);
            this.asuntosTableAdapter3.Fill(this.abogados3DataSet1.Asuntos);

        }

        private void Estilo()
        {
            dataGridView1.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter; // Centrar encabezados
            dataGridView1.ColumnHeadersDefaultCellStyle.Font = new Font("Segoe UI", 10,
FontStyle.Bold); // Negritas en encabezados
            dataGridView1.ColumnHeadersDefaultCellStyle.ForeColor =
ColorTranslator.FromHtml("#212121"); // Color de texto de encabezados
```

```
            dataGridView1.EnableHeadersVisualStyles = false; // Desactivar estilos visuales
para aplicar el estilo personalizado
            dataGridView1.ColumnHeadersDefaultCellStyle.BackColor =
ColorTranslator.FromHtml("#F5F5F5"); // Color de fondo de encabezados

            dataGridView1.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
        }

        private void dataGridView1_RowHeaderMouseClick(object sender,
DataGridViewCellMouseEventArgs e)
        {
            if (e.RowIndex >= 0)
            {
                DataGridViewRow selectedRow = dataGridView1.Rows[e.RowIndex];

                // Obtener el valor de la celda de DNI
                txtDNI.Text = selectedRow.Cells[0].Value.ToString();

                // Manejo seguro para DateTimePicker
                try
                {
                    // Comprobar si el valor es nulo o no puede convertirse
                    if (selectedRow.Cells[1].Value != null &&
DateTime.TryParse(selectedRow.Cells[1].Value.ToString(), out DateTime fecha1))
                    {
                        dateTimePicker2.Value = fecha1;
                    }
                    else
                    {
                        // Manejar el caso en que no se pueda convertir
                        dateTimePicker2.Value = DateTime.Now; // O cualquier valor
predeterminado que consideres
                    }

                    if (selectedRow.Cells[2].Value != null &&
DateTime.TryParse(selectedRow.Cells[2].Value.ToString(), out DateTime fecha2))
                    {
                        dateTimePicker1.Value = fecha2;
```

```
                }
                else
                {
                    dateTimePicker1.Value = DateTime.Now; // Valor predeterminado
                }
            }
            catch (InvalidCastException)
            {
                // Manejar la excepción aquí (por ejemplo, mostrar un mensaje de error)
                MessageBox.Show("Error al convertir la fecha. Verifica los datos en la fila
seleccionada.");
            }

            // Configurar el ComboBox con el valor de la celda
            comboBoxEstado.SelectedItem = selectedRow.Cells[3].Value?.ToString();
        }
    }


    private void button2_Click(object sender, EventArgs e)
    {
        Close();
        Form1 regresraform1 = new Form1();
        regresraform1.Show();
    }

    private void panel1_Paint(object sender, PaintEventArgs e)
    {

    }

    private void btnasignarabogado_Click(object sender, EventArgs e)
    {
        Form6 form6 = new Form6(this); // Pasa la instancia actual de Form4
        form6.Show();
    }

    private void btnasignarcliente_Click(object sender, EventArgs e)
    {
```

```
            Form7 form7 = new Form7(this); // Pasa la instancia actual de Form4
            form7.Show();
        }

        private void txtDNI_KeyPress(object sender, KeyPressEventArgs e)
        {
            txtDNI.MaxLength = 10;
            // Verifica si la tecla presionada es un número o la tecla de retroceso (Backspace)
            if (!char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar))
            {
                e.Handled = true;  // Cancela la entrada si no es un número
            }
        }


    }
}
```

Form 6.cs (Ventana para ver la lista de abogados)

```
using System;
using System.Linq;
using System.Runtime.InteropServices;
using System.Windows.Forms;

namespace Bufete_de_abogados
{
    public partial class Form6 : Form
    {
        private readonly DataClasses1DataContext db;

        private Form4 form4Instancia;

        public Form6(Form4 form4)
        {
            InitializeComponent();
            db = new DataClasses1DataContext();
```

```
            this.form4Instancia = form4;
        }


        private void Form6_Load(object sender, EventArgs e)
        {
            ConfigurarDataGridView();
            ActualizarDatos();
        }

        private void ConfigurarDataGridView()
        {
            dataGridView1.AutoGenerateColumns = true;
            dataGridView1.AllowUserToAddRows = false;
            dataGridView1.AllowUserToDeleteRows = false;
            dataGridView1.MultiSelect = false;
            dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
            dataGridView1.ReadOnly = true;
            dataGridView1.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.Fill;
        }

        private void btn_cerrar_Click(object sender, EventArgs e)
        {
            this.Close(); // Cierra el formulario actual
        }

        private void btn_maximizar_Click(object sender, EventArgs e)
        {
            this.WindowState = FormWindowState.Maximized;
            btn_maximizar.Visible = false;
            btn_restaurar.Visible = true;
        }

        private void btn_restaurar_Click(object sender, EventArgs e)
        {
            this.WindowState = FormWindowState.Normal;
            btn_restaurar.Visible = false;
            btn_maximizar.Visible = true;
```

```
        }

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

        //con  esto se mueve  la ventana
        [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 Barra_Titulo_MouseDown(object sender, MouseEventArgs e)
        {
            ReleaseCapture();
            SendMessage(this.Handle, 0x112, 0xf012, 0);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            try
            {
                if (string.IsNullOrWhiteSpace(busquedapornombretxt.Text))
                {
                    MessageBox.Show("Por favor ingrese un nombre para buscar.",
                                    "Advertencia",
                                    MessageBoxButtons.OK,
                                    MessageBoxIcon.Warning);
                    return;
                }

                // Usar el procedimiento almacenado mediante LINQ to SQL
                var resultados =
db.BuscarProcuradoresPorNombre(busquedapornombretxt.Text).ToList();
                if (resultados.Count == 0)
                {
                    MessageBox.Show("No se encontraron procuradores con ese nombre.",
```

```
"Información", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            else
            {
                dataGridView1.DataSource = resultados;
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Error al buscar procuradores: {ex.Message}",
                        "Error",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Error);
        }

    }

    private void ActualizarDatos()
    {
        try
        {
            // Mostrar todos los procuradores
            var todosLosProcuradores = db.Procuradores.ToList();
            dataGridView1.DataSource = todosLosProcuradores;
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Error al cargar los datos: {ex.Message}",
                        "Error",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Error);
        }
    }

    private void dataGridView1_SelectionChanged(object sender, EventArgs e)
    {

    }
```

```
        private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
        {
            if (e.RowIndex >= 0)
            {
                var filaSeleccionada = dataGridView1.Rows[e.RowIndex];
                var procuradorID = filaSeleccionada.Cells[0].Value.ToString();

                // Envía el ProcuradorID al Form4
                form4Instancia.SetProcuradorID(procuradorID);

                // También puedes cerrar Form6 si ya no es necesario después de seleccionar
el ID
                this.Close();
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {

        }
    }
}
```

Form 7.cs (Ventana para ver la lista de clientes)

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

namespace Bufete_de_abogados
```

```
{
    public partial class Form7 : Form
    {
        private readonly DataClasses1DataContext db;

        private Form4 form4Instancia;

        public Form7(Form4 form4)
        {
            InitializeComponent();
            db = new DataClasses1DataContext();
            this.form4Instancia = form4;
        }

        private void Form7_Load(object sender, EventArgs e)
        {
            ConfigurarDataGridView();
            ActualizarDatos();
        }

        private void ConfigurarDataGridView()
        {
            dataGridView1.AutoGenerateColumns = true;
            dataGridView1.AllowUserToAddRows = false;
            dataGridView1.AllowUserToDeleteRows = false;
            dataGridView1.MultiSelect = false;
            dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
            dataGridView1.ReadOnly = true;
            dataGridView1.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.Fill;
        }

        private void btn_cerrar_Click(object sender, EventArgs e)
        {
            this.Close(); // Cierra el formulario actual
        }

        private void btn_maximizar_Click(object sender, EventArgs e)
        {
```

```
            this.WindowState = FormWindowState.Maximized;
            btn_maximizar.Visible = false;
            btn_restaurar.Visible = true;
        }

        private void btn_restaurar_Click(object sender, EventArgs e)
        {
            this.WindowState = FormWindowState.Normal;
            btn_restaurar.Visible = false;
            btn_maximizar.Visible = true;
        }

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

        //con  esto se mueve  la ventana
        [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 Barra_Titulo_MouseDown(object sender, MouseEventArgs e)
        {
            ReleaseCapture();
            SendMessage(this.Handle, 0x112, 0xf012, 0);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            try
            {
                if (string.IsNullOrWhiteSpace(busquedapornombretxt.Text))
                {
                    MessageBox.Show("Por favor ingrese un nombre para buscar.",
                                    "Advertencia",
                                    MessageBoxButtons.OK,
```

```
                    MessageBoxIcon.Warning);
                return;
            }

            var resultados =
db.BuscarClientesPorNombre(busquedapornombretxt.Text).ToList();
            if (resultados.Count == 0)
            {
                MessageBox.Show("No se encontraron clientes con ese nombre.",
"Información", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            else
            {
                dataGridView1.DataSource = resultados;
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Error al buscar clientes: {ex.Message}",
                    "Error",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
        }

    }

    private void ActualizarDatos()
    {
        try
        {
            var todosLosClientes = db.Clientes.ToList();
            dataGridView1.DataSource = todosLosClientes;
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Error al cargar los datos: {ex.Message}",
                    "Error",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
```

```
            }
        }

        private void dataGridView1_SelectionChanged(object sender, EventArgs e)
        {

        }

        private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
        {
            if (e.RowIndex >= 0)
            {
                var filaSeleccionada = dataGridView1.Rows[e.RowIndex];
                var dniCliente = filaSeleccionada.Cells[0].Value.ToString();

                // Envía el DNI del cliente al Form4
                form4Instancia.SetClienteDNI(dniCliente);

                // Cierra el formulario actual si ya no es necesario después de seleccionar el ID
                this.Close();
            }
        }
    }
}
```

Archivo SQL para el programa de abogados

```
CREATE DATABASE Abogados3
GO
USE Abogados3
GO

-------------------------------------PARTE DE CLIENTES-----------------------------------------
CREATE TABLE Clientes (
    DNI_clientes VARCHAR(50) PRIMARY KEY,
    Nombre VARCHAR(50),
    Apellido VARCHAR(50),
    Direccion VARCHAR(50),
    TelefonoCliente VARCHAR(50)
);
GO

-- PROCEDIMIENTO para ingresar datos a la tabla clientes
CREATE PROCEDURE ingresar_clientes
    @DNI_clientes VARCHAR(50),
    @Nombre VARCHAR(50),
    @Apellido VARCHAR(50),
    @Direccion VARCHAR(50),
    @TelefonoCliente VARCHAR(50)
AS
BEGIN
    INSERT INTO Clientes (DNI_clientes, Nombre, Apellido, Direccion, TelefonoCliente)
    VALUES (@DNI_clientes, @Nombre, @Apellido, @Direccion, @TelefonoCliente);
END;
GO

-- PROCEDIMIENTO para leer datos de la tabla Clientes
CREATE PROCEDURE LEER_ingresar_clientes
AS
BEGIN
    SELECT * FROM Clientes;
END;
GO

-- PROCEDIMIENTO para borrar con ID(DNI)
CREATE PROCEDURE Borrar_ingresar_clientes
```

```
    @DNI_clientes VARCHAR(50)
AS
BEGIN
    DELETE FROM Clientes
    WHERE DNI_clientes = @DNI_clientes;
END;
GO

-- PROCEDIMIENTO PARA ACTUALIZAR la tabla Cliente
CREATE PROCEDURE ActualizarCliente(
    @DNI_clientes VARCHAR(50),
    @NuevoNombre VARCHAR(50),
    @NuevoApellido VARCHAR(50),
    @NuevaDireccion VARCHAR(50),
    @NuevoTelefonoCliente VARCHAR(50)
)
AS
BEGIN
    UPDATE Clientes
    SET Nombre = @NuevoNombre,
        Apellido = @NuevoApellido,
        Direccion = @NuevaDireccion,
        TelefonoCliente = @NuevoTelefonoCliente
    WHERE DNI_clientes = @DNI_clientes;
END;
GO

----------------------------------------------PARTE DE
PROCURADORES-------------------------------------------------------------
-- Tabla de Procuradores
CREATE TABLE Procuradores (
    ProcuradorID VARCHAR(50) PRIMARY KEY,
    Nombre VARCHAR(50),
    Apellido VARCHAR(50),
    Email VARCHAR(50),
    TelefonoProcurador VARCHAR(50)
);
GO

-- PROCEDIMIENTO para ingresar datos a la tabla Procuradores
CREATE PROCEDURE ingresar_Procuradores
```

```
    @ProcuradorID VARCHAR(50),
    @Nombre VARCHAR(50),
    @Apellido VARCHAR(50),
    @Email VARCHAR(50),
    @TelefonoProcurador VARCHAR(50)
AS
BEGIN
    INSERT INTO Procuradores (ProcuradorID, Nombre, Apellido, Email, TelefonoProcurador)
    VALUES (@ProcuradorID, @Nombre, @Apellido, @Email, @TelefonoProcurador);
END;
GO

-- PROCEDIMIENTO para borrar con ID(ProcuradorID)
CREATE PROCEDURE Borrar_Procuradores
    @ProcuradorID VARCHAR(50)
AS
BEGIN
    DELETE FROM Procuradores
    WHERE ProcuradorID = @ProcuradorID;
END;
GO

-- PROCEDIMIENTO PARA ACTUALIZAR la tabla PROCURADOR
CREATE PROCEDURE ActualizarProcurador(
    @ProcuradorID VARCHAR(50),
    @NuevoNombre VARCHAR(50),
    @NuevoApellido VARCHAR(50),
    @NuevoEmail VARCHAR(50),
    @NuevoTelefonoProcurador VARCHAR(50)
)
AS
BEGIN
    UPDATE Procuradores
    SET Nombre = @NuevoNombre,
        Apellido = @NuevoApellido,
        Email = @NuevoEmail,
        TelefonoProcurador = @NuevoTelefonoProcurador
    WHERE ProcuradorID = @ProcuradorID;
END;
GO
```

```
  ----------------------------------------------PARTE DE
ASUNTOS-------------------------------------------------------------
  CREATE TABLE Asuntos (
      AsuntoID VARCHAR(50) PRIMARY KEY,
      DNI_clientes VARCHAR(50),
      FechaInicio DATE,
      FechaFinal DATE,
      Estado VARCHAR(50),
      DatosCaso VARCHAR(MAX),
      FOREIGN KEY (DNI_clientes) REFERENCES Clientes(DNI_clientes) ON DELETE NO
ACTION
  );
  GO

  -- PROCEDIMIENTO para ingresar datos a la tabla Asuntos
  CREATE PROCEDURE ingresar_Asuntos
      @AsuntoID VARCHAR(50),
      @DNI_clientes VARCHAR(50),
      @FechaInicio DATE,
      @FechaFinal DATE,
      @Estado VARCHAR(50),
      @DatosCaso VARCHAR(MAX)
  AS
  BEGIN
      INSERT INTO Asuntos (AsuntoID, DNI_clientes, FechaInicio, FechaFinal, Estado,
DatosCaso)
      VALUES (@AsuntoID, @DNI_clientes, @FechaInicio, @FechaFinal, @Estado,
@DatosCaso);
  END;
  GO

  drop procedure ingresar_Asuntos
  -- PROCEDIMIENTO para borrar con ID(AsuntoID)
  CREATE PROCEDURE Borrar_Asunto
      @AsuntoID VARCHAR(50)
  AS
  BEGIN
      DELETE FROM Asuntos
      WHERE AsuntoID = @AsuntoID;
  END;
  GO
```

```
-- PROCEDIMIENTO PARA ACTUALIZAR la tabla Asuntos
CREATE PROCEDURE ActualizarAsunto(
    @AsuntoID VARCHAR(50),
    @FechaInicio DATE,
    @FechaFinal DATE,
    @Estado VARCHAR(50),
    @DatosCaso VARCHAR(MAX)
)
AS
BEGIN
    UPDATE Asuntos
    SET FechaInicio = @FechaInicio,
        FechaFinal = @FechaFinal,
        Estado = @Estado,
        DatosCaso = @DatosCaso
    WHERE AsuntoID = @AsuntoID;  -- Agregar condición para la actualización
END;
GO

-- Procedimientos adicionales
CREATE PROCEDURE BuscarProcuradoresPorNombre
    @NombreIngresado VARCHAR(50)
AS
BEGIN
    SELECT *
    FROM Procuradores
    WHERE Nombre LIKE '%' + @NombreIngresado + '%';
END;
GO

CREATE PROCEDURE BuscarClientesPorNombre
    @NombreIngresado VARCHAR(50)
AS
BEGIN
    SELECT *
    FROM Clientes
    WHERE Nombre LIKE '%' + @NombreIngresado + '%';
END;
GO
```

```
CREATE PROCEDURE ObtenerTodosLosClientes
AS
BEGIN
   SELECT * FROM Clientes;
END;
GO

CREATE PROCEDURE BorrarClienteConVerificacion
   @DNI_clientes VARCHAR(50)
AS
BEGIN
   -- Verificar si el cliente tiene asuntos asociados
   IF EXISTS (SELECT 1 FROM Asuntos WHERE DNI_clientes = @DNI_clientes)
   BEGIN
      RAISERROR('No se puede borrar el cliente porque tiene asuntos asociados.', 16, 1);
      RETURN;
   END;

   -- Si no hay asuntos, proceder a borrar el cliente
   DELETE FROM Clientes
   WHERE DNI_clientes = @DNI_clientes;

   PRINT 'Cliente borrado exitosamente.';
END;
GO
```

Plan de Pruebas

Objetivo del Plan de Pruebas

El propósito de este plan es garantizar que el sistema cumpla con los requisitos funcionales y no funcionales, asegurando su calidad, rendimiento, seguridad y usabilidad.

Alcance de las Pruebas

Se probarán los siguientes módulos y funcionalidades:

- Registro, modificación y eliminación de clientes
- Registro y administración de abogados (procuradores)
- Asignación de casos
- Visualización de datos
- Validaciones en formularios
- Mensajes de error y retroalimentación
- Seguridad en el acceso y restricciones

Tipos de Pruebas

Tipo de prueba	Descripción
Unitarias	Validan funciones individuales como el guardado o edición de registros.
Integración	Aseguran la correcta interacción entre formularios y base de datos.
Sistema	Evalúan el sistema en su conjunto simulando el flujo real.
Aceptación	Pruebas realizadas desde la perspectiva del usuario final (abogado o asistente).

Casos de Prueba

ID	Caso de Prueba	Datos de Entrada	Resultado Esperado
C P01	Registrar cliente válido	Todos los campos completos y correctos	Cliente guardado exitosamente
C P02	DNI duplicado	DNI ya existente en la BD	Mensaje de error: "DNI duplicado"
C P03	Campos vacíos en cliente	Formulario sin llenar	Mensaje de error por campos vacíos
C P04	Registrar abogado sin correo válido	Correo sin “@”	Mensaje: "Correo inválido"
C P05	Teléfono con letras (cliente)	“664abc1234”	Mensaje: "Teléfono debe contener solo números"
C P06	Asignación de caso con fechas incorrectas	Fecha de inicio > fecha fin	Mensaje de advertencia: "Fecha inválida"

C P07	Eliminar cliente con caso activo	Cliente con caso “En curso”	Error: “No se puede eliminar cliente con asunto activo”
C P08	Eliminar procurador con ID inválido	ID inexistente	Mensaje de error: "No se encontró procurador"
C P09	Cierre del sistema desde el menú principal	Click en botón "Salir"	Aplicación cerrada correctamente
C P10	Cambiar entre formularios	Navegación por menú	Formulario cambia sin errores

Criterios de Aceptación

●El 100% de los casos de prueba críticos deben pasar.

●Los errores detectados deben ser corregidos antes del despliegue.

●La validación de datos debe impedir entradas inválidas o repetidas.

Recursos Necesarios

●Equipo de pruebas: programador y evaluador

●Entorno de prueba:

○Sistema operativo: Windows 10/11

○Software: Visual Studio, SQL Server

○Herramientas: Explorador de bases de datos, depurador de Visual Studio

Conclusión

El sistema “Tijuana Legalis” cumple con los objetivos establecidos, ofreciendo una solución eficiente para la gestión de casos legales en bufetes de abogados. La implementación de una interfaz intuitiva, una base de datos robusta y funcionalidades específicas mejora la organización, seguridad y accesibilidad de la información. El proyecto demuestra la viabilidad de aplicar ingeniería de software para resolver problemas del ámbito jurídico, con potencial para futuras mejoras como soporte multiusuario o integración con plataformas móviles.

Referencias

●Microsoft. (n.d.). *Windows Forms overview*. Microsoft Learn. Recuperado el 13 de abril de 2025, de https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/

●Zarza Morales, J. D., Rojas Osuna, J. J., & Sebastián Martínez García. (2025). *Documentación funcional y técnica del sistema Tijuana Legalis* [Manuscrito no publicado]. Instituto Tecnológico de Tijuana.

●Microsoft. (2023). *Documentación de C#*. Recuperado de https://learn.microsoft.com/es-es/dotnet/csharp/

●Microsoft. (2023). *Documentación de SQL Server*. Recuperado de https://learn.microsoft.com/es-es/sql/