

ASTROSOFT

# Documentación de ASTRA

Jose Angel Villa Ramírez

Ámbar Azul Ronquillo López

Iker Samuel Requenes Hernández

Cristian Daniel Campa Padilla

Documentación general del desarrollo de la aplicación de escritorio ASTRA (Agenda Saludable con Tecnología para la Regulación de Atención médica).

# Índice

<b>Documentación del Proyecto ASTRA.....</b>	<b>4</b>
<b>1. Descripción del Proyecto.....</b>	<b>4</b>
<b>2. Objetivo del Proyecto .....</b>	<b>4</b>
<b>3. Requerimientos Funcionales (RF).....</b>	<b>4</b>
<b>4. Requerimientos No Funcionales (RNF) .....</b>	<b>5</b>
<b>5. Tecnologías Utilizadas .....</b>	<b>5</b>
<b>6. Arquitectura del Sistema .....</b>	<b>5</b>
<b>7. Diagramas UML .....</b>	<b>5</b>
<b>8. Modelado de Base de Datos.....</b>	<b>7</b>
<b>9. Indicadores del Proyecto.....</b>	<b>7</b>
<b>10. Indicadores de Desempeño de la Aplicación.....</b>	<b>8</b>
<b>11. Desarrollo Iterativo e Incremental .....</b>	<b>8</b>
<b>12. Mejora Continua.....</b>	<b>8</b>
<b>13. Evidencias Visuales y Capturas de Pantalla .....</b>	<b>9</b>
<b>Formulario de inicio de sesión.....</b>	<b>13</b>
<b>Código completo.....</b>	<b>13</b>
<b>Diseño.....</b>	<b>19</b>
<b>Iniciar sesión .....</b>	<b>19</b>
<b>Cancelar .....</b>	<b>21</b>
<b>Iniciar sesión aquí.....</b>	<b>21</b>
<b>Formulario de registro de usuarios.....</b>	<b>22</b>
<b>Código completo.....</b>	<b>22</b>
<b>Diseño.....</b>	<b>25</b>
<b>Registrar .....</b>	<b>25</b>
<b>Formulario general de los datos de los pacientes.....</b>	<b>26</b>
<b>Código completo.....</b>	<b>26</b>
<b>Diseño.....</b>	<b>28</b>
<b>Metodo de cargar pacientes .....</b>	<b>28</b>
<b>Agregar paciente .....</b>	<b>30</b>

<b>Eliminar paciente .....</b>	30
<b>Agendar y modificar cita .....</b>	32
<b>Eliminar cita .....</b>	33
<b>Expediente .....</b>	34
<b>Modificar paciente .....</b>	34
<b>Formulario agregar paciente.....</b>	35
<b>Código completo.....</b>	35
<b>Diseño.....</b>	39
<b>Agregar paciente .....</b>	39
<b>Formulario de agendar/modificar una cita .....</b>	41
<b>Código completo:.....</b>	41
<b>Diseño.....</b>	45
<b>Agendar/Modificar cita .....</b>	45
<b>Formulario de expediente .....</b>	47
<b>Código completo.....</b>	47
<b>Diseño.....</b>	50
<b>Expediente .....</b>	50
<b>Confirmar expediente .....</b>	50
<b>Modificar pacientes .....</b>	51
<b>Código completo.....</b>	51
<b>Base de datos .....</b>	56
<b>MySQL.....</b>	56
<b>Tablas .....</b>	59

# **Documentación del Proyecto ASTRA**

---

## **1. Descripción del Proyecto**

El proyecto ASTRA consiste en el desarrollo de una aplicación de escritorio desarrollada con C# y Windows Forms, diseñada para la gestión de citas y pacientes en un centro comunitario. El sistema busca informatizar el proceso de control de pacientes y consultas, permitiendo una administración más eficiente, organizada y segura.

---

## **2. Objetivo del Proyecto**

El objetivo principal de ASTRA es registrar pacientes y agendar citas médicas. Como objetivo secundario, se incorpora la funcionalidad de redactar y almacenar un expediente clínico correspondiente a cada cita registrada.

---

## **3. Requerimientos Funcionales (RF)**

### **Código**

RF-01 Registro de citas medicas

RF-02 Visualizar vista general de pacientes

RF-03 Gestionar pacientes

RF-04 Modificar y eliminar citas

RF-05 Mostrar hora de la citas

RF-06 Ingreso con usuario y contraseña

RF-07 Búsqueda de pacientes

RF-08 Control de citas no permitir citas en el mismo horario

RF-09 Confirmar contraseña

---

## 4. Requerimientos No Funcionales (RNF)

### Código Requerimiento No Funcional Descripción Detallada

RNF-01 Interfaz intuitiva	UX/UI sencillo, formularios cortos.
RNF-02 Rendimiento	Tiempo de respuesta eficaz.
RNF-03 Funcionalidad offline	Agendar citas sin conexión y sincronizar luego.
RNF-06 Compatibilidad	Ejecutable en equipos con recursos limitados.
RNF-07 Diseño moderno	Estética médica, tipografía clara.

---

## 5. Tecnologías Utilizadas

- Lenguaje: C#
- Entorno: Windows Forms
- Base de Datos: MySQL

---

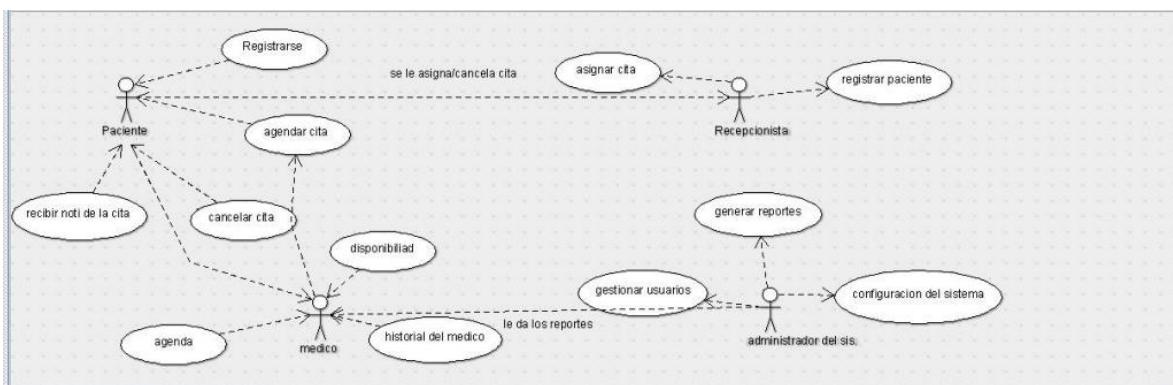
## 6. Arquitectura del Sistema

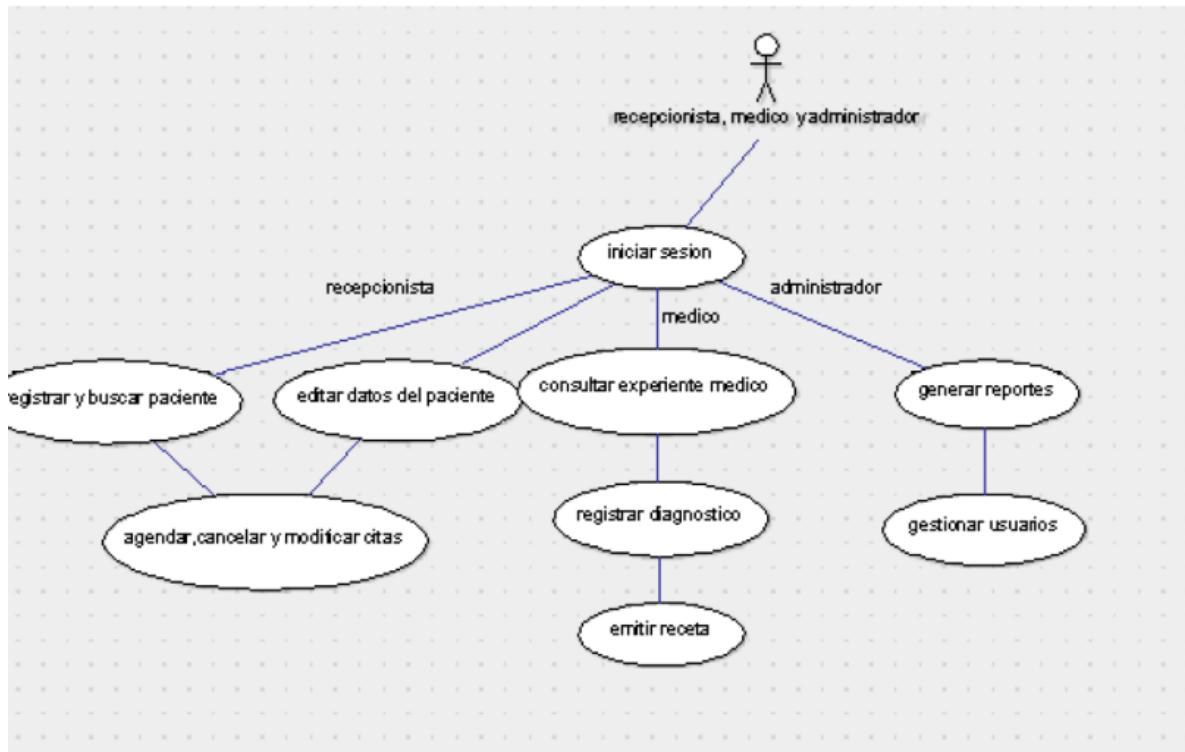
*Se realizaron los ajustes de la aplicación para que sea funcional en arquitecturas x32 y x64 bits respectivamente.*

---

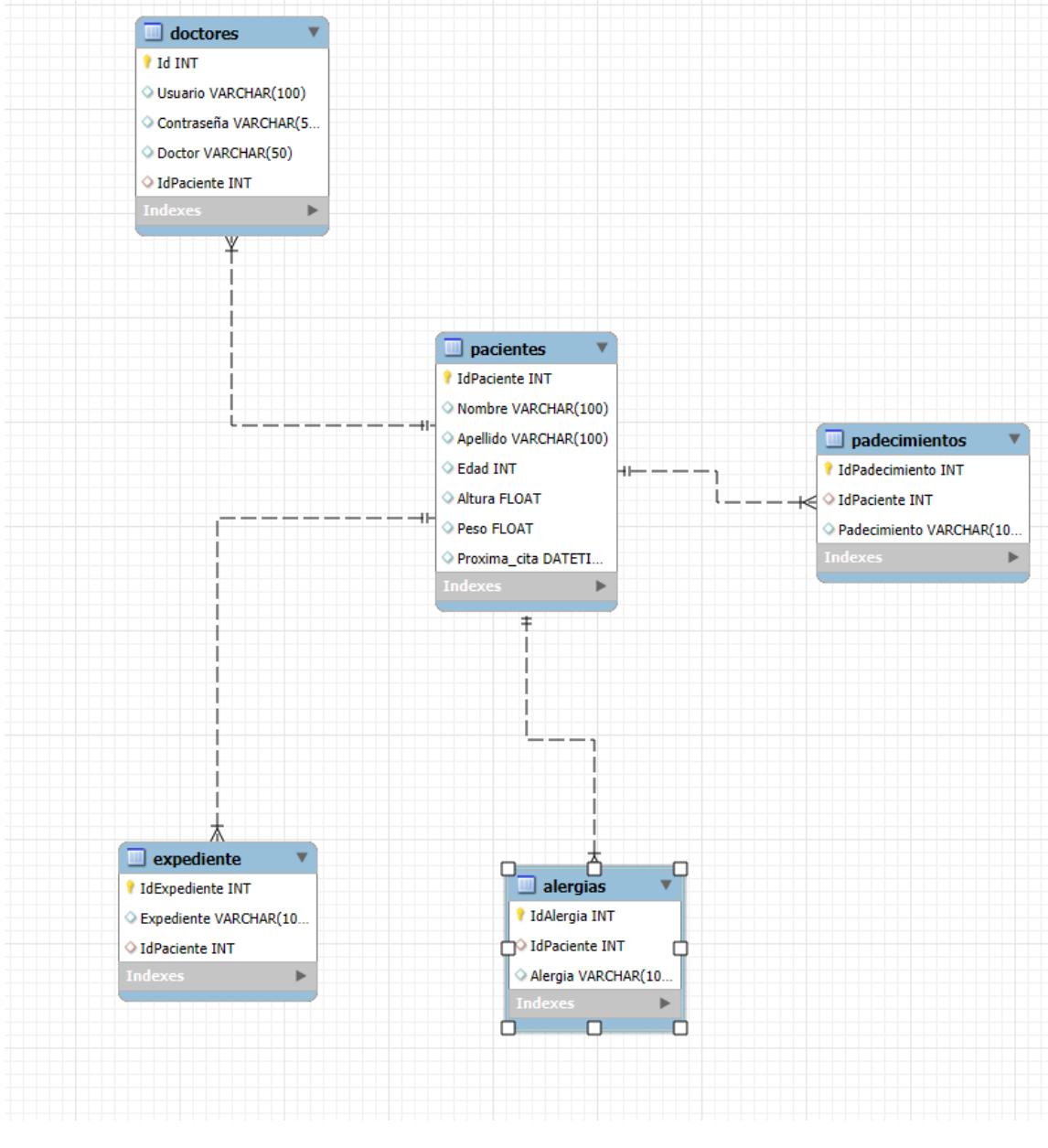
## 7. Diagramas UML

*Estos diagramas son meramente ilustrativos y pueden tener variaciones con respecto al producto final.*





## 8. Modelado de Base de Datos



## 9. Indicadores del Proyecto

**Tiempo de desarrollo:** 3 meses

**Costo estimado:+**

*Costos estimados por persona, no representan los costos totales del desarrollo.*

- Equipo de cómputo: \$4000 MXN
- Licencias: \$1200 MXN

- Otros gastos: \$400 MXN (transporte, energía, internet, oficinas)

#### **Alcance:**

- Prototipo monousuario
  - Base de datos local integrada
  - Cumplimiento del objetivo de agendamiento de citas
- 

## **10. Indicadores de Desempeño de la Aplicación**

*Aceptación de los usuarios/interesados*

*Promedios generales*

- *Satisfacción: 4.5 / 5*
- *Probabilidad de recomendar: 3.7 / 5*
- *Calificación general: 4.0 / 5*

*Esto indica una aceptación positiva en general, aunque todavía hay espacio para mejorar en la recomendación y la percepción global.*  
).

---

## **11. Desarrollo Iterativo e Incremental**

El proyecto se desarrolló en fases:

1. Inicio de sesión y registro de usuarios
2. Diseño de interfaz principal con DataGridView para pacientes
3. Ventanas independientes para registro de pacientes y citas
4. Integración del expediente por paciente
5. Módulos de eliminación y edición

Cada iteración fue evaluada y validada antes de continuar con la siguiente, lo que permitió implementar mejoras constantes sobre el prototipo inicial.

---

## **12. Mejora Continua**

Se implementó una metodología de retroalimentación continua:

- Validación funcional tras cada módulo
  - Ajustes según experiencia de usuario
  - Revisiones internas de código
  - Priorización de funcionalidades clave antes de implementar extras
- 

### 13. Evidencias Visuales y Capturas de Pantalla

*ASTRA Gestor de citas médicas*

Usuario

Contraseña   Mostrar contraseña

¿No está registrado? [Regístrate aquí!](#)

Iniciar sesión Cancelar

*ASTRA Gestor de citas médicas*

Doctor(a)

Usuario  \*

Contraseña  \*

Confirmé su contraseña por favor

Mostrar contraseña

Registrar Cancelar

Doctor(a): Angel Villa

	ID	Nombre	Apellido	Edad	Altura	Peso	Alergias	Padecimientos	Proxima cita	Hora
▶	1	Angel	Villa	19	1.65	65	Ninguno	Ninguno		
◀										

 Agregar Paciente  
 Eliminar Paciente  
 Modificar Datos Paciente  
 Agendar Cita  
[redacted]  
Buscar paciente  
 Modificar Cita  
 Eliminar Cita  
 Acceder al Expediente  
 Cerrar sesión



Nombre

Apellido

Edad

Altura (Metros)

Peso (Kilogramos)



Alergias medicinales

Padecimientos



Confirmar

X

Paciente	Angel
Edad	Alergias
<input type="text"/>	<input type="text"/>
Altura (Metros)	Padecimientos
<input type="text"/>	<input type="text"/>
Peso (Kilogramos)	
<input type="text"/>	

**Datos previos**

Edad	Altura (Metros)	Peso (Kilogramos)
19	1.65	65
Alergias	Padecimientos	
<input type="text"/>	<input type="text"/>	
Ninguno	Ninguno	

\*

  
**Confirmar**

X

- Cita nueva

Seleccione la fecha y hora de la cita a agendar

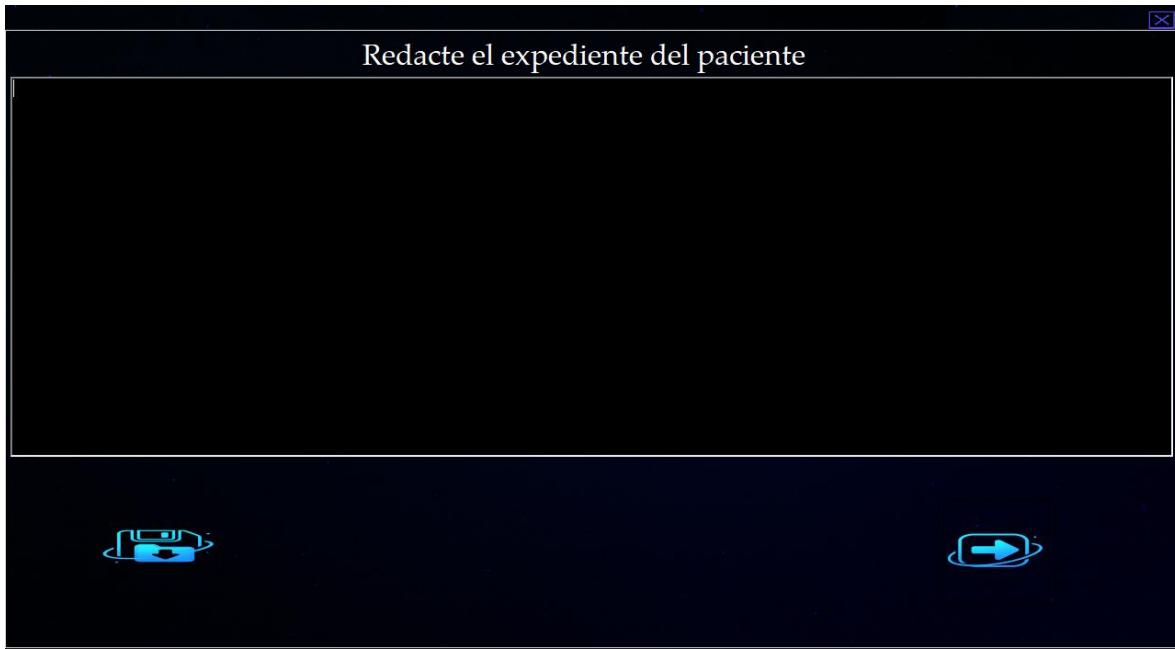
«	agosto de 2025	»				
lun	mar	mié	jue	vie	sáb	dom
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

Hoy: 28/8/2025

Ingrese la hora formato 12 horas HH:mm

21:36





## Formulario de inicio de sesión

### Código completo

```
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 Microsoft.Data.SqlClient;
using MySql.Data.MySqlClient;
using System.IO;
namespace Astra
{
    public partial class Form1 : Form
    {
        string rutaDb;
        string cadenaConexion;
        public Form1()
        {

            InitializeComponent();

            //Cadena de conexión

            // string ruta = Path.Combine(Application.StartupPath,
            @"Data\AstraDB.mdf");
        }
    }
}
```

```

        cadenaConexion =
"Server=localhost;Database=Astra;Uid=root;Pwd=277353;";

    }

private void Base()
{
    using(MySqlConnection conn = new MySqlConnection(cadenaConexion))
    {
        try
        {
            conn.Open();

            string scriptDb = @"

                CREATE DATABASE IF NOT EXISTS Astra;";
            new MySqlCommand(scriptDb, conn).ExecuteNonQuery();

            string script = @"
                USE Astra;
                -- archivo: astra_schema.sql
                CREATE DATABASE IF NOT EXISTS `Astra`
                    DEFAULT CHARACTER SET utf8mb4
                    DEFAULT COLLATE utf8mb4_general_ci;
                USE `Astra`;

                -- Tabla Pacientes (creada primero porque las
demás referencian su PK)
                CREATE TABLE IF NOT EXISTS `Pacientes` (
                    `IdPaciente` INT NOT NULL AUTO_INCREMENT,
                    `Nombre` VARCHAR(100) NOT NULL,
                    `Apellido` VARCHAR(100) NOT NULL,
                    `Edad` INT NULL,
                    `Altura` FLOAT NULL,
                    `Peso` FLOAT NULL,
                    `Proxima_cita_Fecha` DATETIME NULL,
                    `Hora` TIME NULL,
                    PRIMARY KEY (`IdPaciente`)
                ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

                -- Tabla Doctores (Id AUTO_INCREMENT + FK a
Pacientes, IdPaciente NULLABLE)
                CREATE TABLE IF NOT EXISTS `Doctores` (
                    `Id` INT NOT NULL AUTO_INCREMENT,
                    `Usuario` VARCHAR(100) NOT NULL,
                    `Contraseña` VARCHAR(255) NOT NULL, -- mantenido
tal como en tu código; se recomienda usar hash
                    `Doctor` VARCHAR(50) NULL,
                    `IdPaciente` INT NULL,
                    PRIMARY KEY (`Id`),
                    UNIQUE KEY `UK_Doctor_Usuario` (`Usuario`),
                )
            ";
        }
    }
}

```

```

CONSTRAINT `FK_Doctores_Pacientes` FOREIGN KEY
(`IdPaciente`)
    REFERENCES `Pacientes` (`IdPaciente`)
    ON DELETE SET NULL
    ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Tabla Alergias
CREATE TABLE IF NOT EXISTS `Alergias` (
`IdAlergia` INT NOT NULL AUTO_INCREMENT,
`IdPaciente` INT NOT NULL,
`Alergia` VARCHAR(100) NULL,
PRIMARY KEY (`IdAlergia`),
KEY `idx_alergias_paciente` (`IdPaciente`),
CONSTRAINT `FK_Alergias_Pacientes` FOREIGN KEY
(`IdPaciente`)

REFERENCES `Pacientes` (`IdPaciente`)
ON DELETE CASCADE
ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Tabla Padecimientos
CREATE TABLE IF NOT EXISTS `Padecimientos` (
`IdPadecimiento` INT NOT NULL AUTO_INCREMENT,
`IdPaciente` INT NOT NULL,
`Padecimiento` VARCHAR(100) NULL,
PRIMARY KEY (`IdPadecimiento`),
KEY `idx_padecimientos_paciente` (`IdPaciente`),
CONSTRAINT `FK_Padecimientos_Pacientes` FOREIGN
KEY (`IdPaciente`)

REFERENCES `Pacientes` (`IdPaciente`)
ON DELETE CASCADE
ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Tabla Expediente
CREATE TABLE IF NOT EXISTS `Expediente` (
`IdExpediente` INT NOT NULL AUTO_INCREMENT,
`Expediente` VARCHAR(100) NULL,
`IdPaciente` INT NOT NULL,
PRIMARY KEY (`IdExpediente`),
KEY `idx_expediente_paciente` (`IdPaciente`),
CONSTRAINT `FK_Expediente_Pacientes` FOREIGN KEY
(`IdPaciente`)

REFERENCES `Pacientes` (`IdPaciente`)
ON DELETE CASCADE
ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

";

MySqlCommand cmd = new MySqlCommand(script, conn);
cmd.ExecuteNonQuery();
}
catch(Exception ex)
{
    MessageBox.Show(ex.Message);
}

```

```

        }

    }

}

private void ProbarConexion()
{
    using (MySqlConnection conn = new
MySqlConnection(cadena_conexion))
    {
        try
        {
            conn.Open();
            MessageBox.Show("☑ Conexión exitosa a MySQL");
        }
        catch (Exception ex)
        {
            MessageBox.Show("☒ Error al conectar: " + ex.Message);
        }
    }
}

private void Form1_Load(object sender, EventArgs e)
{
    ProbarConexion();
    Base();
}

private void btnIniciarSesion_Click(object sender, EventArgs e)
{
    //Creamos las variables para traer el texto de los textbox
    string usuario = txtUsuario.Text;
    string contraseña = txtContraseña.Text;

    using (MySqlConnection con = new
MySqlConnection(cadena_conexion))
    {
        try // Es fundamental tener un try-catch que envuelva la
apertura de la conexión y las operaciones
        {
            con.Open();

            // Si tu tabla 'Usuarios' es la misma que 'usuarios' en
el ejemplo de Form4,
                // la capitalización podría importar dependiendo de la
configuración de la DB,
                    // pero generalmente SQLite no distingue
mayúsculas/minúsculas para nombres de tablas/columnas.

            // --- CORRECCIÓN DE LA CONSULTA Y LOS PARÁMETROS ---
            // 1. Uso de parámetros nombrados (@usuario, @contraseña)
- MUY RECOMENDADO
            string consulta = "SELECT COUNT(*) FROM Doctores WHERE
Usuario = @Usuario AND Contraseña = @Contraseña";

```

```

        using (MySqlCommand comando = new MySqlCommand(consulta,
con))
    {
        // 2. Añadir parámetros por nombre, esto es más
robusto y claro.
        comando.Parameters.AddWithValue("@Usuario", usuario);
// El nombre del parámetro debe coincidir con la consulta
        comando.Parameters.AddWithValue("@Contraseña",
contraseña);

        // --- CORRECCIÓN EN LA CONVERSIÓN DE EXECUTESCALAR --
-- 
        // ExecuteScalar devuelve un 'object'. Para COUNT(*),
será un 'long' (0 o más).
        // Es más seguro usar Convert.ToInt32() o un cast a
'long' primero.
        object resultado = comando.ExecuteScalar();
        int cuenta = 0;

        if (resultado != null && resultado != DBNull.Value)
        {
            cuenta = Convert.ToInt32(resultado); // Convierte
el resultado a int de forma segura
        }
        // Si resultado es null o DBNull.Value (lo cual no
debería ocurrir con COUNT(*)), cuenta seguirá siendo 0.

        if (cuenta > 0)
        {
            MessageBox.Show("Inicio de sesión exitoso");

            Form3 form3 = new Form3(usuario);
            form3.Show();
            this.Hide();
        }
        else
        {
            MessageBox.Show("Usuario y/o contraseña
incorrectos."); // Mensaje más preciso
        }
    }
    catch (MySqlException ex)
    { // Captura errores específicos de SQL
        MessageBox.Show($"Error de base de datos: {ex.Message}",
"Error de Conexión", MessageBoxButtons.OK, MessageBoxIcon.Error);
        // Opcional: Debug.WriteLine(ex.ToString()); para ver más
detalles en la salida de depuración
    }
    catch (Exception ex) // Captura cualquier otro error
inesperado
    {
        MessageBox.Show($"Ocurrió un error inesperado:
{ex.Message}", "Error General", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    // El 'using' se encarga de cerrar la conexión
automáticamente al salir del bloque.
}

```

```

        }

        //Envia al formulario de registro
        private void linkLabel1_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
        {
            txtUsuario.Clear();
            txtContraseña.Clear();
            Form2 form2 = new Form2();
            form2.Show();

        }

        private void btnCancelar_Click(object sender, EventArgs e)
        {
            txtUsuario.Clear();
            txtContraseña.Clear();
        }

        private void Salir_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
        //Variables para arrastrar y definir una localizacion de inicio de la
pantalla

        private bool arrastrar = false;
        private Point puntoInicio;
        //Evento para soltar fijar la localizacion de la ventana al soltar el
click
        private void panel3_MouseDown(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
            {
                arrastrar = true;
                puntoInicio = new Point(e.X, e.Y);
            }
        }
        //evento para mover la localizacion de la ventana al mover el mouse
mientras el boton este sostenido
        private void panel3_MouseMove(object sender, MouseEventArgs e)
        {
            if(arrastrar)
            {
                Point p = PointToScreen(e.Location);
                this.Location = new Point(p.X - puntoInicio.X, p.Y -
puntoInicio.Y);
            }
        }

        //evento para cambiar la variable de arrastrar a falso
        private void panel3_MouseUp(object sender, MouseEventArgs e)
        {
            arrastrar = false;
        }
        //Permite ver la contraseña
        private void checkBox1_CheckedChanged(object sender, EventArgs e)
    
```

```

    {
        if (checkBox1.Checked)
        {
            txtContraseña.PasswordChar = '\0';
        }
        else
        {
            txtContraseña.PasswordChar = '*';
        }
    }

}

```

## Diseño



## Iniciar sesión

Configuramos el evento de click en el botón de iniciar previamente ya tenemos establecida una ruta y una cadena de conexión a nuestra base de datos sql, por lo que el evento de click manda a llamar a los datos de los campos de texto pertenecientes al usuario y a la contraseña para así realizar una búsqueda a través de la tabla, siguiendo la lógica si retorna un valor menor a 0, es decir negativo entonces quiere decir que no se encontró de ninguno modo registrado nuestro usuario y muestra un mensaje diciendo que el usuario y/o la

contraseña son incorrectos, en el caso de retornar un valor mayor a 0 le permite al usuario entrar al menú principal.

```
private void btnIniciarSesion_Click(object sender, EventArgs e)
{
    //Creamos las variables para traer el texto de los textbox
    string usuario = txtUsuario.Text;
    string contraseña = txtContraseña.Text;

    using (MySqlConnection con = new MySqlConnection(cadena_conexion))
    {
        try // Es fundamental tener un try-catch que envuelva la apertura de la conexión y las operaciones
        {
            con.Open();

            // Si tu tabla 'Usuarios' es la misma que 'usuarios' en el ejemplo de Form4,
            // la capitalización podría importar dependiendo de la configuración de la DB,
            // pero generalmente SQLite no distingue mayúsculas/minúsculas para nombres de tablas/columnas.

            // --- CORRECCIÓN DE LA CONSULTA Y LOS PARÁMETROS ---
            // 1. Uso de parámetros nombrados (@usuario, @contraseña) - MUY RECOMENDADO
            string consulta = "SELECT COUNT(*) FROM Doctores WHERE Usuario = @Usuario AND Contraseña = @Contraseña";

            using (MySqlCommand comando = new MySqlCommand(consulta, con))
            {
                // 2. Añadir parámetros por nombre, esto es más robusto y claro.
                comando.Parameters.AddWithValue("@Usuario", usuario); // El nombre del parámetro debe coincidir con la consulta
                comando.Parameters.AddWithValue("@Contraseña", contraseña);

                // --- CORRECCIÓN EN LA CONVERSIÓN DE EXECUTESCALAR ---
                // ExecuteScalar devuelve un 'object'. Para COUNT(*), será un 'long' (0 o más).
                // Es más seguro usar Convert.ToInt32() o un cast a 'long' primero.
                object resultado = comando.ExecuteScalar();
                int cuenta = 0;

                if (resultado != null && resultado != DBNull.Value)
                {
                    cuenta = Convert.ToInt32(resultado); // Convierte el resultado a int de forma segura
                }
                // Si resultado es null o DBNull.Value (lo cual no debería ocurrir con COUNT(*)), cuenta seguirá siendo 0.

                if (cuenta > 0)
                {
                    MessageBox.Show("Inicio de sesión exitoso");
                }
            }
        }
    }
}
```

```

        Form3 form3 = new Form3(usuario);
        form3.Show();
        this.Hide();
    }
    else
    {
        MessageBox.Show("Usuario y/o contraseña incorrectos.");
// Mensaje más preciso
    }
}
catch (MySqlException ex)
{
    // Captura errores específicos de SQL
    MessageBox.Show($"Error de base de datos: {ex.Message}", "Error de Conexión", MessageBoxButtons.OK, MessageBoxIcon.Error);
    // Opcional: Debug.WriteLine(ex.ToString()); para ver más detalles en la salida de depuración
}
catch (Exception ex) // Captura cualquier otro error inesperado
{
    MessageBox.Show($"Ocurrió un error inesperado: {ex.Message}", "Error General", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
// El 'using' se encarga de cerrar la conexión automáticamente al salir del bloque.
}
}

```

## Cancelar

El evento de click para el botón cancelar es mas sencillo pues no realiza ningún ajuste directo en la base de datos ya que solo limpia usando la palabra reservada clear los campos de texto.

```

private void btnCancelar_Click(object sender, EventArgs e)
{
    txtUsuario.Clear();
    txtContraseña.Clear();
}

```

## Iniciar sesión aquí

Este label de enlace permite visualizar un formulario para que el usuario registre un usuario y contraseña para acceder a la aplicación, haciendo uso de un insert a la base de datos y de buscar que no se relacionen los valores de los campos con uno ya existente permite registrar exitosamente al usuario.

```

private void linkLabel1_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
{
    Form2 form2 = new Form2();
    form2.Show();
}

```

```
}
```

## Formulario de registro de usuarios

### Código completo

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
//using Microsoft.Data.SqlClient;
using MySql.Data.MySqlClient;

namespace Astra
{

    public partial class Form2 : Form
    {
        string ruta;
        string cadena_conexion;

        public Form2()
        {
            InitializeComponent();

            // string ruta = Path.Combine(Application.StartupPath,
            @"Data\AstraDB.mdf");
            cadena_conexion =
"Server=localhost;Database=Astra;Uid=root;Pwd=277353;";

        }

        private void btnRegistrar_Click(object sender, EventArgs e)
        {
            string nombre = txtNombre.Text;
            string usuario = txtRegistroUsuario.Text;
            string contraseña = txtRegistroContraseña.Text;
            string confirm = txtConfirmarContraseña.Text;

            if (usuario == "" || contraseña == "" || nombre == "")
            {
```

```

        MessageBox.Show("Por favor ingrese todos los datos");
        return;
    }

    if(confirm != contraseña)
    {
        MessageBox.Show("Las contraseñas no coinciden ",
"Advertencia", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    using (MySqlConnection con = new
MySqlConnection(cadena_conexion))
    {
        try
        {
            con.Open();
            string consulta = "SELECT COUNT(*) FROM Doctores WHERE
Usuario = @Usuario";
            MySqlCommand verificar = new MySqlCommand(consulta, con);
            verificar.Parameters.AddWithValue("@Usuario", usuario);
            long existe = (long)verificar.ExecuteScalar();
            if (existe > 0)
            {

                MessageBox.Show("El usuario ya esta registrado");
                return;
            }
            //Insertar el nuevo usuario
            string insertar = "INSERT INTO Doctores (Usuario,
Contraseña, Doctor) VALUES (@Usuario,@Contraseña, @Doctor)";
            MySqlCommand cmd = new MySqlCommand(insertar, con);
            cmd.Parameters.AddWithValue("@Usuario", usuario);
            cmd.Parameters.AddWithValue("@Contraseña", contraseña);
            cmd.Parameters.AddWithValue("@Doctor", nombre);
            cmd.ExecuteNonQuery();

            MessageBox.Show("Usuario registrado con exito");
            this.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error al registrar " + ex.Message);
        }
    }
}

private void Cerrar_Click(object sender, EventArgs e)
{
    this.Hide();
}

private bool arrastrar = false;
private Point puntoInicio;
private void panel3_MouseDown(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left)
    {

```

```
        arrastrar = true;
        puntoInicio = new Point(e.X, e.Y);
    }
}
private void panel3_MouseMove(object sender, MouseEventArgs e)
{
    if (arrastrar)
    {
        Point p = PointToScreen(e.Location);
        this.Location = new Point(p.X - puntoInicio.X, p.Y -
puntoInicio.Y);
    }
}
private void panel3_MouseUp(object sender, MouseEventArgs e)
{
    arrastrar = false;
}

private void btnCancelar_Click(object sender, EventArgs e)
{
    this.Hide();
}

private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox1.Checked)
    {
        txtRegistroContraseña.PasswordChar = '\0';
        txtConfirmarContraseña.PasswordChar = '\0';
    }
    else
    {
        txtRegistroContraseña.PasswordChar= '*';
        txtConfirmarContraseña.PasswordChar = '*';
    }
}
private void label2_Click(object sender, EventArgs e)
{
}
}
```

## Diseño



## Registrar

Al igual que al iniciar sesión se toman los datos de las cajas de texto pertenecientes a usuario y contraseña de tal manera que se vuelve a realizar un insert en la en nuestra base de datos detectando que no exista un valor registrado que coincida con lo ingresado por el usuario. De manera que se hace un select que retorna la cantidad de usuarios registrados en nuestra base de datos y haciendo uso de nuestro sql command verificar va a ir retomando cada valor de la columna, es decir si buscamos un usuario “admin” va retomando cada valor registrado hasta que coincide con nuestro valor ingresado en este ejemplo es “admin” si coincide retorna un valor de 1 y si no existe retorna 0 entonces si existe y es mayor a 0 mostrara un mensaje que ya esta registrado, en el caso de que no se realiza un insert a la base de datos para agregar el nuevo valor.

```
private void btnRegistrar_Click(object sender, EventArgs e)
{
    string nombre = txtNombre.Text;
    string usuario = txtRegistroUsuario.Text;
    string contraseña = txtRegistroContraseña.Text;
    string confirm = txtConfirmarContraseña.Text;

    if (usuario == "" || contraseña == "" || nombre == "")
    {
        MessageBox.Show("Por favor ingrese todos los datos");
        return;
    }

    if(confirm != contraseña)
    {
        MessageBox.Show("Las contraseñas no coinciden ", "Advertencia",
        MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}
```

```

        return;
    }

    using (MySqlConnection con = new MySqlConnection(cadena_conexion))
    {
        try
        {
            con.Open();
            string consulta = "SELECT COUNT(*) FROM Doctores WHERE Usuario = @Usuario";
            MySqlCommand verificar = new MySqlCommand(consulta, con);
            verificar.Parameters.AddWithValue("@Usuario", usuario);
            long existe = (long)verificar.ExecuteScalar();
            if (existe > 0)
            {

                MessageBox.Show("El usuario ya esta registrado");
                return;
            }
            //Insertar el nuevo usuario
            string insertar = "INSERT INTO Doctores (Usuario, Contraseña, Doctor) VALUES (@Usuario,@Contraseña, @Doctor)";
            MySqlCommand cmd = new MySqlCommand(insertar, con);
            cmd.Parameters.AddWithValue("@Usuario", usuario);
            cmd.Parameters.AddWithValue("@Contraseña", contraseña);
            cmd.Parameters.AddWithValue("@Doctor", nombre);
            cmd.ExecuteNonQuery();

            MessageBox.Show("Usuario registrado con exito");
            this.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error al registrar " + ex.Message);
        }
    }
}

```

## Formulario general de los datos de los pacientes

### Código completo

```

private void btnRegistrar_Click(object sender, EventArgs e)
{
    string nombre = txtNombre.Text;
    string usuario = txtRegistroUsuario.Text;
    string contraseña = txtRegistroContraseña.Text;
    string confirm = txtConfirmarContraseña.Text;

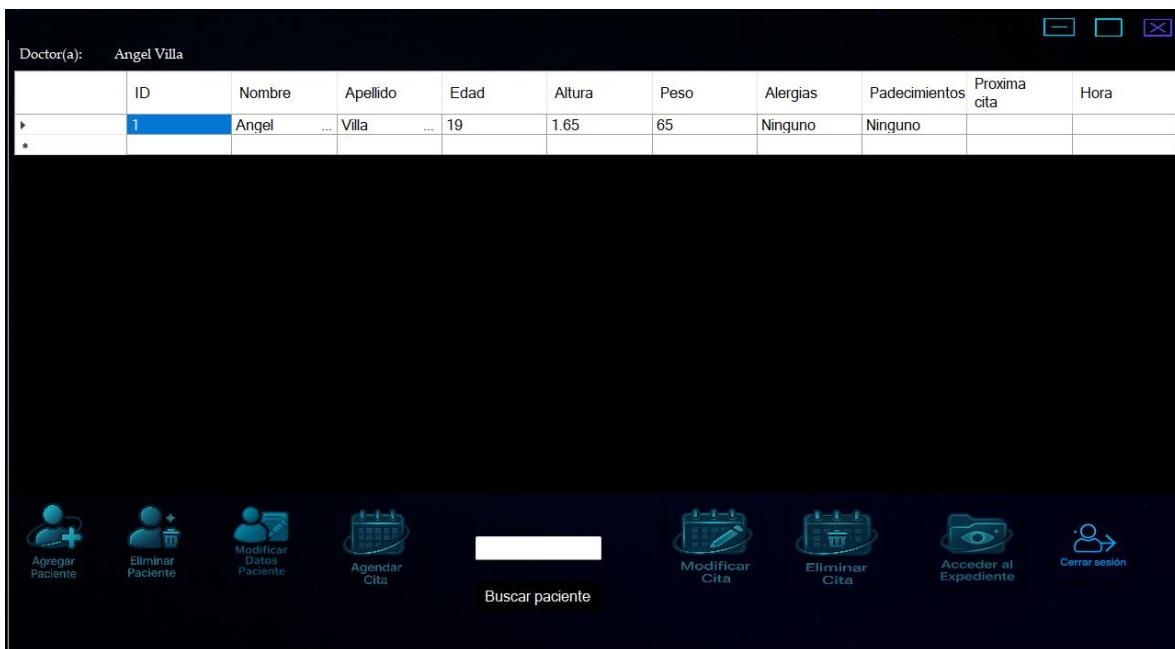
    if (usuario == "" || contraseña == "" || nombre == "")
    {
        MessageBox.Show("Por favor ingrese todos los datos");
        return;
    }

    if(confirm != contraseña)

```

```
{  
    MessageBox.Show("Las contraseñas no coinciden ", "Advertencia",  
    MessageBoxButtons.OK, MessageBoxIcon.Warning);  
    return;  
}  
  
using (MySqlConnection con = new MySqlConnection(cadena_conexion))  
{  
    try  
    {  
        con.Open();  
        string consulta = "SELECT COUNT(*) FROM Doctores WHERE Usuario =  
@Usuario";  
        MySqlCommand verificar = new MySqlCommand(consulta, con);  
        verificar.Parameters.AddWithValue("@Usuario", usuario);  
        long existe = (long)verificar.ExecuteScalar();  
        if (existe > 0)  
        {  
  
            MessageBox.Show("El usuario ya esta registrado");  
            return;  
        }  
        //Insertar el nuevo usuario  
        string insertar = "INSERT INTO Doctores (Usuario, Contraseña,  
Doctor) VALUES (@Usuario,@Contraseña, @Doctor)";  
        MySqlCommand cmd = new MySqlCommand(insertar, con);  
        cmd.Parameters.AddWithValue("@Usuario", usuario);  
        cmd.Parameters.AddWithValue("@Contraseña", contraseña);  
        cmd.Parameters.AddWithValue("@Doctor", nombre);  
        cmd.ExecuteNonQuery();  
  
        MessageBox.Show("Usuario registrado con exito");  
        this.Close();  
    }  
    catch (Exception ex)  
    {  
        MessageBox.Show("Error al registrar " + ex.Message);  
    }  
}
```

## Diseño



### Método de cargar pacientes

Al momento de abrir el formulario existe un evento de carga que manda a llamar a este método de carga de pacientes, lo que hace este método es escribir las columnas de nuestro datagridview del mismo modo en el que se encuentran en la tabla de la que vamos a proceder a realizar un select y vamos a usar un datareader, entonces primero se establece la conexión a la base de datos y obtenemos una consulta de los datos registrados a través de un select, después de ejecutar la consulta creamos y ejecutamos un datareader (lector de datos) y también vamos a crear una datatable, este funciona como una pequeña tabla y el usando el metodo loadreader se carga toda la información leída por datareader a la tabla y posteriormente esto mismo se vuelve a cargar esta vez en el datagridview.

```
private void CargarPacientes()
{
    cadena_conexion = "Server=localhost;Database=Astra;Uid=root;Pwd=277353;";

    using (MySqlConnection con = new MySqlConnection(cadena_conexion))
    {
        try
        {
            con.Open();

            // 1. Cargar pacientes
            string consultaPacientes = "SELECT IdPaciente AS ID, Nombre,
Apellido, Edad, Altura, Peso FROM Pacientes";
            DataTable tablaPacientes = new DataTable();
            using (MySqlDataAdapter adaptador = new
MySqlDataAdapter(consultaPacientes, con))
            {
                adaptador.Fill(tablaPacientes);
            }
        }
    }
}
```

```

        }

        // 2. Crear columnas adicionales
        if (!tablaPacientes.Columns.Contains("Alergias"))
            tablaPacientes.Columns.Add("Alergias", typeof(string));

        if (!tablaPacientes.Columns.Contains("Padecimientos"))
            tablaPacientes.Columns.Add("Padecimientos", typeof(string));

        if (!tablaPacientes.Columns.Contains("Proxima cita"))
            tablaPacientes.Columns.Add("Proxima cita", typeof(DateTime));

        if (!tablaPacientes.Columns.Contains("Hora"))
            tablaPacientes.Columns.Add("Hora", typeof(TimeSpan));

        // 3. Rellenar datos relacionados
        foreach (DataRow fila in tablaPacientes.Rows)
        {
            int idPaciente = Convert.ToInt32(fila["ID"]);

            // --- Alergias ---
            using (MySqlCommand cmdAlergias = new MySqlCommand(
                "SELECT Alergia FROM Alergias WHERE IdPaciente =
@IdPaciente", con))
            {
                cmdAlergias.Parameters.AddWithValue("@IdPaciente",
idPaciente);
                using (MySqlDataAdapter da = new
MySqlDataAdapter(cmdAlergias))
                {
                    DataTable tAlergias = new DataTable();
                    da.Fill(tAlergias);
                    fila["Alergias"] = string.Join(", ",
tAlergias.AsEnumerable().Select(r =>
r["Alergia"].ToString()));
                }
            }

            // --- Padecimientos ---
            using (MySqlCommand cmdPadecimientos = new MySqlCommand(
                "SELECT Padecimiento FROM Padecimientos WHERE IdPaciente =
@IdPaciente", con))
            {
                cmdPadecimientos.Parameters.AddWithValue("@IdPaciente",
idPaciente);
                using (MySqlDataAdapter da = new
MySqlDataAdapter(cmdPadecimientos))
                {
                    DataTable tPadecimientos = new DataTable();
                    da.Fill(tPadecimientos);
                    fila["Padecimientos"] = string.Join(", ",
tPadecimientos.AsEnumerable().Select(r =>
r["Padecimiento"].ToString()));
                }
            }

            // --- Próxima cita + hora ---
            using (MySqlCommand cmdCita = new MySqlCommand(

```

```

        "SELECT Proxima_cita_Fecha, Hora FROM Pacientes WHERE
IdPaciente = @IdPaciente", con))
    {
        cmdCita.Parameters.AddWithValue("@IdPaciente",
idPaciente);
        using (MySqlDataReader dr = cmdCita.ExecuteReader())
        {
            if (dr.Read())
            {
                fila["Proxima cita"] = dr["Proxima_cita_Fecha"]
!= DBNull.Value ? dr["Proxima_cita_Fecha"] : DBNull.Value;
                fila["Hora"] = dr["Hora"] != DBNull.Value ?
dr["Hora"] : DBNull.Value;
            }
        }
    }

    // 4. Mostrar en DataGridView
    dgvPacientes.DataSource = tablaPacientes;
}
catch (Exception ex)
{
    MessageBox.Show("Error al cargar pacientes: " + ex.Message);
}
}
}

```

## Agregar paciente

Simplemente mandamos a llamar a nuestro formulario de agregar paciente pero hay que hacer unos ajustes importantes, dentro de nuestro formulario de agregar pacientes nos vamos a encontrar con un evento llamado “PacienteAgregado” por lo que es muy importante hacer uso de la suscripción del evento de paciente agregar desde este formulario. Y utilizamos showdialog para retornar hacia este formulario los datos de las variables que encontramos en el formulario de agregar paciente.

```

private void btnAgregarPaciente_Click(object sender, EventArgs e)
{
    Form4 = new Form4();
    //Suscribirse al evento
    form4.PacienteAgregado += () =>
    {
        CargarPacientes();
    };
    form4.ShowDialog();
}

```

## Eliminar paciente

Para eliminar un paciente se hace directo del formulario principal entonces no es necesario llamar a otro formulario, simplemente se selecciona el paciente a eliminar y se extra el valor de la celda de id del paciente, en este caso al extraer un valor se hace de tipo de string entonces se vuelve a convertir a tipo int, una vez hecho esto abrimos la conexión a la base

de datos y ejecutamos un delete desde la tabla de usuarios Si el id de paciente es igual al id de paciente que extraemos y convertimos a tipo int y finalmente volvemos a llamar al método para recargar la información de los pacientes que aun quedan registrados .

```
private void button2_Click(object sender, EventArgs e)
{
    if(dgvPacientes.SelectedRows.Count == 0)
    {
        MessageBox.Show("Por favor seleccione un paciente a eliminar");
        return;
    }
    DialogResult confirmacion = MessageBox.Show("¿Esta seguro de que desea
eliminar al paciente? ", "Confirmar eliminacion ", MessageBoxButtons.YesNo,
MessageBoxIcon.Warning);

    if (confirmacion == DialogResult.No)
        return;

    int idpaciente = Convert.ToInt32
    (dgvPacientes.SelectedRows[0].Cells["ID"].Value);

    using (MySqlConnection conn = new MySqlConnection(cadena_conexion))
    {
        try
        {
            conn.Open();
            //Al usar claves foraneas el orden de eliminacion es importante
            string eliminarPadecimientos = "DELETE FROM Padecimientos WHERE
IdPaciente = @IdPaciente";
            MySqlCommand cmdPadecimientos = new
MySqlCommand(eliminarPadecimientos, conn);
            cmdPadecimientos.Parameters.AddWithValue("@IdPaciente",
idpaciente);
            cmdPadecimientos.ExecuteNonQuery();

            string eliminarAlergias = "DELETE FROM Alergias WHERE IdPaciente
= @IdPaciente";
            MySqlCommand cmdAlergias = new MySqlCommand(eliminarAlergias,
conn);
            cmdAlergias.Parameters.AddWithValue("@IdPaciente", idpaciente);
            cmdAlergias.ExecuteNonQuery();

            string eliminarPaciente = "DELETE FROM Pacientes WHERE
IdPaciente = @IdPaciente";
            MySqlCommand cmd = new MySqlCommand(eliminarPaciente, conn);
            cmd.Parameters.AddWithValue("@IdPaciente", idpaciente);
            cmd.ExecuteNonQuery();

            MessageBox.Show("Paciente eliminado correctamente");
        }
        catch (Exception ex)
```

```

    {
        MessageBox.Show("No se pudo eliminar el paciente " +
ex.Message);
    }
}
CargarPacientes();
}

```

## Agendar y modificar cita

Estos 2 formularios siguen el mismo principio lógico del formulario que agrega paciente debemos seleccionar un paciente, conseguir el valor de la celda de id y posteriormente mandamos a llamar al formulario de agenda y modificación de cita incorporando el valor extraído de id y suscribimos al evento de agregarcita.

```

private void btnAgendar_Click(object sender, EventArgs e)
{
    if (dgvPacientes.SelectedRows.Count == 0)
    {
        MessageBox.Show("Seleccione un paciente para agendar su cita por favor");
        return;
    }
    int id =
int.Parse(dgvPacientes.CurrentRow.Cells["IdPaciente"].Value.ToString());
    Form5 form5 = new Form5(id);
    form5.CitaAgregada += () =>
    {
        CargarPacientes();
    };

    form5.ShowDialog();
}

private void btnModificar_Click(object sender, EventArgs e)
{
    if (dgvPacientes.SelectedRows.Count == 0)
    {
        MessageBox.Show("Seleccione un paciente para agendar su cita por favor");
        return;
    }
    int id =
int.Parse(dgvPacientes.CurrentRow.Cells["ID"].Value.ToString());
    Form5 form5 = new Form5(id);
    form5.CitaAgregada += () =>
    {
        CargarPacientes();
    };

    form5.ShowDialog();
}

```

```
}
```

## Eliminar cita

Comenzamos obteniendo un valor de la celda de próxima cita, si no existe un valor entonces mandamos un mensaje de que no tiene una cita, y si existe mandamos a extraer ese valor para castearlo a tipo int y asignarlo a una variable, entramos nuevamente a la base de datos y ejecutamos un update para actualizar el valor de la cita directamente en la columna de próxima cita como un valor null y finalmente mostramos un mensaje de confirmación y volvemos a llamar al evento de cargar paciente.

```
private void btnEliminar_Click(object sender, EventArgs e)
    {//Variables para obtener los valores de las celdas de acuerdo a su tipo de
    //dato: Fecha y numerico

        object valor = dgvPacientes.CurrentRow.Cells["Proxima cita"].Value;
        object valorhora = dgvPacientes.CurrentRow.Cells["Hora"].Value;
        if (string.IsNullOrEmpty(valor.ToString()))
        {
            MessageBox.Show("No hay cita agendada, error al eliminar");
            return;
        }
        if (string.IsNullOrEmpty(valorhora.ToString()))
        {
            MessageBox.Show("No hay una hora registrada, error al eliminar");
        }

        DialogResult confirmacion = MessageBox.Show("¿Esta seguro de que desea
eliminar la cita del paciente?", "Confirmar eliminacion",
MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
        if (confirmacion == DialogResult.No)
            return;
        int idpaciente =
int.Parse(dgvPacientes.CurrentRow.Cells["ID"].Value.ToString());
        //Variable de nuestra ruta de datos

        using (MySqlConnection con = new MySqlConnection(cadena_conexion))
        {
            try
            {
                con.Open();
                //Actualizacion de la base de datos
                string update = "UPDATE Pacientes SET Proxima_cita_Fecha = NULL,
Hora = NULL WHERE IdPaciente = @IdPaciente";
                //Comando para agregar valores y actualizar
                MySqlCommand cmd = new MySqlCommand(update, con);

                cmd.Parameters.AddWithValue("@IdPaciente", idpaciente);

                cmd.ExecuteNonQuery();
                MessageBox.Show("Cita eliminada correctamente ");
            }
        }
    }
```

```

        catch (Exception ex)
        {
            MessageBox.Show("Conexion fallida " + ex.Message);
        }
    }
    CargarPacientes();
}

```

## Expediente

El uso de este botón es igual al de agregar paciente en el sentido de que solo manda a llamar un formulario nuevo para redactar nuestro expediente haciendo uso del valor del id del paciente del que queremos redactar o revisar un expediente.

```

private void btnExpediente_Click(object sender, EventArgs e)
{
    int id = 0;

    if (dgvPacientes.SelectedRows.Count > 0)
    {
        id =
int.Parse(dgvPacientes.CurrentRow.Cells["ID"].Value.ToString());
    }
    else
    {
        MessageBox.Show("No existe ningun paciente " +
MessageBoxIcon.Error);
    }

    if(id == 0)
    {
        MessageBox.Show("Seleccione un paciente para ver su expediente");
        return;
    }
    Form6 form6 = new Form6(id);

    form6.ShowDialog();
}

```

## Modificar paciente

Este botón abre un nuevo form que permite realizar nuevos ajustes a un paciente ya existente

```

private void button1_Click(object sender, EventArgs e)
{
    int id = 0;

    if (dgvPacientes.SelectedRows.Count > 0)
    {
        id =
int.Parse(dgvPacientes.CurrentRow.Cells["ID"].Value.ToString());
    }
}

```

```

    else
    {
        MessageBox.Show("No existe ningun paciente " +
    MessageBoxIcon.Error);
    }

    if (id == 0)
    {
        MessageBox.Show("Seleccione un paciente para modificar su
información");
        return;
    }
    Form7 form7 = new Form7(id);
    form7.PacienteActualizado += () =>
    {
        CargarPacientes();
    };
    form7.ShowDialog();
}

```

## Formulario agregar paciente

### Código completo

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Security.Cryptography;
using System.Security.Policy;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using MySql.Data.MySqlClient;
namespace Astra
{
    public partial class Form4 : Form
    {

        string cadena_conexion;

        public event Action PacienteAgregado; // Evento personalizado

        public class Paciente
        {
            public string Nombre { get; set; }
            public string Apellidos { get; set; }
            public int Edad { get; set; }
        }
    }
}

```

```

        public double Altura { get; set; }
        public double Peso { get; set; }
        public string Alergia { get; set; }
        public string Padecimiento { get; set; }

    }
    public Form4()
    {
        InitializeComponent();

        string ruta = Path.Combine(Application.StartupPath,
 @"Data\AstraDB.mdf");
        cadenaConexion =
"Server=localhost;Database=Astra;Uid=root;Pwd=277353;";

    }
    private void btnAgregar_Click_1(object sender, EventArgs e)
{
    //Obtener variables y datos en los atributos de la clase Paciente
    Paciente paciente = new Paciente();
    try
    {

        paciente.Nombre = txtNombre.Text;
        paciente.Apellidos = txtApellidos.Text;

        paciente.Edad = int.Parse(txtEdad.Text);
        paciente.Altura = double.Parse(txtAltura.Text);
        paciente.Peso = double.Parse(txtPeso.Text);
        paciente.Alergia = txtAlergias.Text;
        paciente.Padecimiento = txtPadecimientos.Text;

        if (paciente.Edad <= 0 || paciente.Edad >= 100 )
        {
            MessageBox.Show("Ingrese una edad valida ",
"Advertencia", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return;
        }

    }
    catch
    {
        MessageBox.Show("Por favor, complete todos los campos
correctamente.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return; // Salir del método si hay un error
    }

    //usando la base de datos
    using (MySqlConnection con = new
MySqlConnection(cadenaConexion))
    {

```

```

try
{
    con.Open();
    //Orden insertar para usar la palabra reservada INSERT INTO para indicar "Insertar en" tabla Pacientes "valores"

        string insertar = @"INSERT INTO Pacientes (Nombre,
Apellido, Edad, Altura, Peso) VALUES (@Nombre, @Apellido, @Edad,
@Altura,@Peso); SELECT
LAST_INSERT_ID();";

        MySqlCommand cmd = new MySqlCommand(insertar,con);
        cmd.Parameters.AddWithValue("@Nombre", paciente.Nombre);
        cmd.Parameters.AddWithValue("@Apellido",
paciente.Apellidos);
        cmd.Parameters.AddWithValue("@Edad", paciente.Edad);
        cmd.Parameters.AddWithValue("@Altura", paciente.Altura);
        cmd.Parameters.AddWithValue("@Peso", paciente.Peso);

        int idpaciente = Convert.ToInt32(cmd.ExecuteScalar());

        //Nueva insercion de los datos de alergias y
padecimientos

        string insertarAlergias = @"INSERT INTO Alergias
(IdPaciente, Alergia) VALUES (@IdPaciente,@Alergia)";

        MySqlCommand cmdAlergias = new
MySqlCommand(insertarAlergias, con);
        cmdAlergias.Parameters.AddWithValue("@IdPaciente",
idpaciente);
        cmdAlergias.Parameters.AddWithValue("@Alergia",
paciente.Alergia);

        string insertarPadecimientos = @"INSERT INTO
Padecimientos (IdPaciente, Padecimiento) VALUES (@IdPaciente,@Padecimiento)";
        MySqlCommand cmdPadecimientos = new
MySqlCommand(insertarPadecimientos, con);
        cmdPadecimientos.Parameters.AddWithValue("@IdPaciente",
idpaciente);
        cmdPadecimientos.Parameters.AddWithValue("@Padecimiento",
paciente.Padecimiento);

        cmdAlergias.ExecuteNonQuery();
        cmdPadecimientos.ExecuteNonQuery();
        MessageBox.Show("Paciente registrado correctamente");
        // Disparar evento para actualizar el otro formulario
        PacienteAgregado?.Invoke();
        this.Close();

    }
    catch (Exception ex)
    {

        MessageBox.Show("Error al registrar paciente" +
ex.Message);
    }
}

```

```
}

private void Cerrar_Click(object sender, EventArgs e)
{
    this.Hide();
}

private bool arrastrar = false;
private Point puntoInicio;
private void panel1_MouseDown(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left)
    {
        arrastrar = true;
        puntoInicio = new Point(e.X, e.Y);
    }
}
private void panel1_MouseMove(object sender, MouseEventArgs e)
{
    if (arrastrar)
    {
        Point p = PointToScreen(e.Location);
        this.Location = new Point(p.X - puntoInicio.X, p.Y -
puntoInicio.Y);
    }
}
private void panel1_MouseUp(object sender, MouseEventArgs e)
{
    arrastrar = false;
}
}
```

## Diseño

Nombre      Apellido      Edad

Altura (Metros)      Peso (Kilogramos)

Alergias medicinales      Padecimientos

Confirmar

## Agregar paciente

Primero junto con la creación de la cadena de conexión a la base de datos debemos crear también un evento de acción public event action de paciente agregado (el mismo al que se suscribió el formulario anterior) y posteriormente vamos a trabajar con una clase paciente para poder hacer el registro, la clase paciente contiene todos los datos necesarios para poder asignarlos como atributos,(Nombre, Apellido, Edad, Altura, Peso, Alergias y Padecimientos. Después del constructor con las conexiones a la base de datos y en el evento de click al botón de agregar paciente vamos a instanciar la clase Paciente creando un objeto paciente y a cada atributo perteneciente le asignamos lo registrado por el usuario en las cajas de texto correspondientes, después de entrar a la base de datos vamos a crear un insert a la tabla de Pacientes y agregamos los valores de cada columna de datos y a través de un

comando tomamos estos valores registrados en los atributos de la clase y los insertamos a la columna correspondiente de la tabla ahora usamos Invoke para disparar el evento agregado de agregar paciente y finalmente mandamos a cerrar esta ventana además de incorporar un catch para obtener cual es el error en el caso de contar con alguno.

```
private void btnAgregar_Click_1(object sender, EventArgs e)
{
    //Obtener variables y datos en los atributos de la clase Paciente
    Paciente paciente = new Paciente();
    try
    {

        paciente.Nombre = txtNombre.Text;
        paciente.Apellidos = txtApellidos.Text;

        paciente.Edad = int.Parse(txtEdad.Text);
        paciente.Altura = double.Parse(txtAltura.Text);
        paciente.Peso = double.Parse(txtPeso.Text);
        paciente.Alergia = txtAlergias.Text;
        paciente.Padecimiento = txtPadecimientos.Text;

        if (paciente.Edad <= 0 || paciente.Edad >= 100 )
        {
            MessageBox.Show("Ingrese una edad valida ", "Advertencia",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return;
        }

    }
    catch
    {
        MessageBox.Show("Por favor, complete todos los campos
correctamente.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return; // Salir del método si hay un error
    }

    //usando la base de datos
    using (MySqlConnection con = new MySqlConnection(cadena_conexion))
    {

        try
        {
            con.Open();
            //Orden insertar para usar la palabra reservada INSERT INTO para
            indicar "Insertar en" tabla Pacientes "valores"

            string insertar = @"INSERT INTO Pacientes (Nombre, Apellido,
Edad, Altura, Peso) VALUES (@Nombre, @Apellido, @Edad,
@Altura,@Peso); SELECT LAST_INSERT_ID();";

            MySqlCommand cmd = new MySqlCommand(insertar,con);
            cmd.Parameters.AddWithValue("@Nombre", paciente.Nombre);
            cmd.Parameters.AddWithValue("@Apellido", paciente.Apellidos);
            cmd.Parameters.AddWithValue("@Edad", paciente.Edad);
            cmd.Parameters.AddWithValue("@Altura", paciente.Altura);
        }
    }
}
```

```

        cmd.Parameters.AddWithValue("@Peso", paciente.Peso);

        int idpaciente = Convert.ToInt32(cmd.ExecuteScalar());

        //Nueva insercion de los datos de alergias y padecimientos

        string insertarAlergias = @"INSERT INTO Alergias (IdPaciente,
Alergia) VALUES (@IdPaciente,@Alergia)";

        MySqlCommand cmdAlergias = new MySqlCommand(insertarAlergias,
con);
        cmdAlergias.Parameters.AddWithValue("@IdPaciente", idpaciente);
        cmdAlergias.Parameters.AddWithValue("@Alergia",
paciente.Alergia);

        string insertarPadecimientos = @"INSERT INTO Padecimientos
(IdPaciente, Padecimiento) VALUES (@IdPaciente,@Padecimiento)";
        MySqlCommand cmdPadecimientos = new
MySqlCommand(insertarPadecimientos, con);
        cmdPadecimientos.Parameters.AddWithValue("@IdPaciente",
idpaciente);
        cmdPadecimientos.Parameters.AddWithValue("@Padecimiento",
paciente.Padecimiento);

        cmdAlergias.ExecuteNonQuery();
        cmdPadecimientos.ExecuteNonQuery();
        MessageBox.Show("Paciente registrado correctamente");
        // Disparar evento para actualizar el otro formulario
        PacienteAgregado?.Invoke();
        this.Close();

    }
    catch (Exception ex)
    {

        MessageBox.Show("Error al registrar paciente" + ex.Message);
    }
}
}

```

## Formulario de agendar/modificar una cita

Código completo:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.OleDb;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using MySql.Data.MySqlClient;

```

```

namespace Astra
{
    public partial class Form5 : Form
    {
        Form1 form1 = new Form1();

        string ruta;
        string cadenaConexion;
        public event Action CitaAgregada;
        private int idpaciente seleccionado;

        public Form5(int IdPaciente)
        {
            InitializeComponent();
            //Direcciones de la base de datos

            //string ruta = Path.Combine(Application.StartupPath,
            @"Data\AstraDB.mdf");
            cadenaConexion =
"Server=localhost;Database=Astra;Uid=root;Pwd=277353;";

            idpaciente seleccionado = IdPaciente;
        }

        private void Form5_Load(object sender, EventArgs e)
        {
            Hora.Format = DateTimePickerFormat.Custom;
            Hora.CustomFormat= "HH:mm";
            Hora.ShowUpDown = true;

        }
        public class Cita
        {
            public DateTime NuevaCita { get; set; }
            public TimeSpan Hora { get; set; }

        }
        private void btnAgendar_Click(object sender, EventArgs e)
        {

            Cita cita = new Cita();
            DateTime fecha;
            fecha = Calendario.SelectionStart;
            TimeSpan hora_cita = new TimeSpan(Hora.Value.Hour,
Hora.Value.Minute, 0);

            cita.NuevaCita = Calendario.SelectionStart;
            TimeSpan horaCita = new TimeSpan(Hora.Value.Hour,
Hora.Value.Minute, 0);
        }
    }
}

```

```

        cita.Hora = horaCita ;




        using (MySqlConnection con = new
MySqlConnection(cadena_conexion))
    {
        try
        {
            con.Open();
            //Insercion a la base de datos mediante un update

            string consultar = "SELECT COUNT(*) FROM Pacientes WHERE
Proxima_cita_Fecha = @Proxima_cita_Fecha AND Hora = @Hora";

            using (MySqlCommand cmd = new MySqlCommand(consultar,
con))
            {
                //Añadir parametros
                cmd.Parameters.AddWithValue("@Proxima_cita_Fecha",
fecha);
                cmd.Parameters.AddWithValue("@Hora", hora_cita);

                object resultado = cmd.ExecuteScalar();
                int cuenta = 0;
                if (resultado != null && resultado != DBNull.Value)
                {
                    cuenta = Convert.ToInt32(resultado);

                }
                if (cuenta > 0)
                {
                    MessageBox.Show("Cita ya agendada porfavor
ingrese otra fecha y/o hora", "advertencia", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                    return;
                }
                else
                {
                    using (MySqlCommand cmd2 = new
MySqlCommand("UPDATE Pacientes SET Proxima_cita_Fecha = @Proxima_cita_Fecha
WHERE IdPaciente = @IdPaciente", con))
                    {

cmd2.Parameters.AddWithValue("@Proxima_cita_Fecha", cita.NuevaCita);
                        cmd2.Parameters.AddWithValue("@IdPaciente",
idpacienteseleccionado);
                        cmd2.ExecuteNonQuery();




                    }
                    using (MySqlCommand cmd2 = new
MySqlCommand("UPDATE Pacientes SET Hora = @Hora WHERE IdPaciente =
@IdPaciente", con))
                    {
                        cmd2.Parameters.AddWithValue("@Hora",
cita.Hora);

```

```

        cmd2.Parameters.AddWithValue("@IdPaciente",
idpacienteseleccionado);
        cmd2.ExecuteNonQuery();

    }

}

}

MessageBox.Show("Fecha y hora de la cita: " +
cita.NuevaCita + " " + cita.Hora);

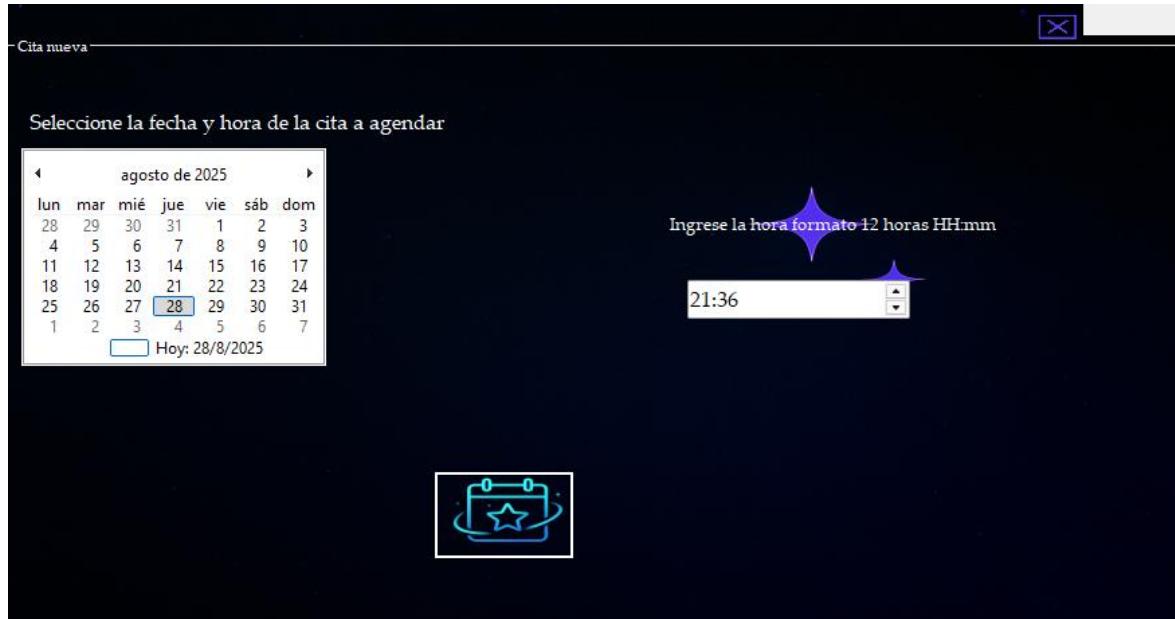
CitaAgregada?.Invoke();
this.Close();
}
catch (Exception ex)
{
    MessageBox.Show("Error al agregar cita nueva " +
ex.Message);
}
}

private void Cerrar_Click(object sender, EventArgs e)
{
    this.Hide();
}
private bool arrastrar = false;
private Point puntoInicio;
private void panel1_MouseDown(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left)
    {
        arrastrar = true;
        puntoInicio = new Point(e.X, e.Y);
    }
}
private void panel1_MouseMove(object sender, MouseEventArgs e)
{
    if (arrastrar)
    {
        Point p = PointToScreen(e.Location);
        this.Location = new Point(p.X - puntoInicio.X, p.Y -
puntoInicio.Y);
    }
}
private void panel1_MouseUp(object sender, MouseEventArgs e)
{
    arrastrar = false;
}

```

```
}
```

## Diseño



## Agendar/Modificar cita

El formulario sigue la misma lógica de crear un evento de acción publica y así mismo este evento debe de ser suscrito al formulario principal entonces seleccionamos un día del calendario y simplemente confirmamos, por lo que en el código dentro de la clase cita y su atributo de nueva cita del tipo DateTime se va a registrar el día seleccionado por el usuario. Por lo que instanciamos un objeto cita perteneciente a la clase Cita y mandamos a llamar el valor seleccionado a través de la propiedad de selectionstart del month calendar y como el valor por defecto de la celda es un valor nulo debemos hacer un update a la base de datos a la columna de próxima cita si la el valor de la columna de id paciente corresponde con el seleccionado anteriormente por lo que es importante tener una variable global que contenga el valor del id del paciente y se guarda a través del constructor del formulario mandando a llamar a la variable que debe estar definida en el formulario anterior, a esto yo le llamo variable de enlace. Finalmente solo hacemos una invocación al evento de acción de cita agregada y cerramos la ventana, también hacemos uso de un catch para poder mostrar el error en caso de tener alguno.

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

```
{
```

```

Cita cita = new Cita();
DateTime fecha;
fecha = Calendario.SelectionStart;
TimeSpan hora_cita = new TimeSpan(Hora.Value.Hour, Hora.Value.Minute, 0);

cita.NuevaCita = Calendario.SelectionStart;
TimeSpan horaCita = new TimeSpan(Hora.Value.Hour, Hora.Value.Minute, 0);
cita.Hora = horaCita ;



using (MySqlConnection con = new MySqlConnection(cadena_conexion))
{
    try
    {
        con.Open();
        //Insercion a la base de datos mediante un update

        string consultar = "SELECT COUNT(*) FROM Pacientes WHERE
Proxima_cita_Fecha = @Proxima_cita_Fecha AND Hora = @Hora";

        using (MySqlCommand cmd = new MySqlCommand(consultar, con))
        {
            //Añadir parametros
            cmd.Parameters.AddWithValue("@Proxima_cita_Fecha", fecha);
            cmd.Parameters.AddWithValue("@Hora", hora_cita);

            object resultado = cmd.ExecuteScalar();
            int cuenta = 0;
            if (resultado != null && resultado != DBNull.Value)
            {
                cuenta = Convert.ToInt32(resultado);

            }
            if (cuenta > 0)
            {
                MessageBox.Show("Cita ya agendada porfavor ingrese otra
fecha y/o hora", "advertencia", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                return;
            }
            else
            {
                using (MySqlCommand cmd2 = new MySqlCommand("UPDATE
Pacientes SET Proxima_cita_Fecha = @Proxima_cita_Fecha WHERE IdPaciente =
@IdPaciente", con))
                {
                    cmd2.Parameters.AddWithValue("@Proxima_cita_Fecha",
cita.NuevaCita);
                    cmd2.Parameters.AddWithValue("@IdPaciente",
idpacienteseleccionado);
                    cmd2.ExecuteNonQuery();

                }
            }
        }
    }
}

```

```

        using (MySqlCommand cmd2 = new MySqlCommand("UPDATE
Pacientes SET Hora = @Hora WHERE IdPaciente = @IdPaciente", con))
        {
            cmd2.Parameters.AddWithValue("@Hora", cita.Hora);
            cmd2.Parameters.AddWithValue("@IdPaciente",
idpacienteseleccionado);
            cmd2.ExecuteNonQuery();

        }

    }

}

MessageBox.Show("Fecha y hora de la cita: " +
cita.NuevaCita + " " + cita.Hora);

CitaAgregada?.Invoke();
this.Close();
}
catch (Exception ex)
{
    MessageBox.Show("Error al agregar cita nueva " + ex.Message);
}
}
}

```

## Formulario de expediente

### Código completo

```

using Microsoft.Data.SqlClient;
using System;
using System.IO;
using System.Windows.Forms;

namespace Astra
{
    public partial class Form6 : Form
    {
        string cadena_conexion;
        public Action pacienteAgregadoi;
        private int idpacienteseleccionado;
        private bool expedienteExiste = false;

        public Form6(int IdPaciente)
        {
            InitializeComponent();
            idpacienteseleccionado = IdPaciente;
        }
    }
}

```

```

        string ruta = Path.Combine(Application.StartupPath,
 @"Data\AstraDB.mdf");
        cadena_conexion = "Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\AstraDB.mdf;
Integrated Security=True;Connect Timeout=30";

    }

private void CargarExpediente()
{
    using (SqlConnection con = new SqlConnection(cadena_conexion))
    {
        try
        {
            con.Open();
            string consulta = "SELECT Expediente FROM Expedientes
WHERE IdExpediente = @IdExpediente";
            SqlCommand cmd = new SqlCommand(consulta, con);
            cmd.Parameters.AddWithValue("@IdExpediente",
idpacienteseleccionado);
            SqlDataReader reader = cmd.ExecuteReader();
            if (reader.Read())
            {
                Expediente.Text = reader["Expediente"].ToString();
                expedienteExiste = true;
            }
            else
            {
                expedienteExiste = false;
                // No puedes dejar el TextBox vacío si es nuevo
                Expediente.Text = "";
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error al cargar el expediente: " +
ex.Message);
        }
    }
}

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

private void btnConfirmar_Click(object sender, EventArgs e)
{
    string expediente = Expediente.Text.Trim();

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

```

```

        }

        using (SqlConnection con = new SqlConnection(cadena_conexion))
        {
            try
            {
                con.Open();

                string query;

                if (expedienteExiste)
                {
                    // Ya existe -> UPDATE
                    query = "UPDATE Expedientes SET Expediente =
@Expediente WHERE IdExpediente = @IdExpediente";
                }
                else
                {
                    // No existe -> INSERT
                    query = "INSERT INTO Expedientes (IdExpediente,
Expediente) VALUES (@IdExpediente, @Expediente)";
                }

                SqlCommand cmd = new SqlCommand(query, con);
                cmd.Parameters.AddWithValue("@Expediente", expediente);
                cmd.Parameters.AddWithValue("@IdExpediente",
idpacienteseleccionado);
                cmd.ExecuteNonQuery();

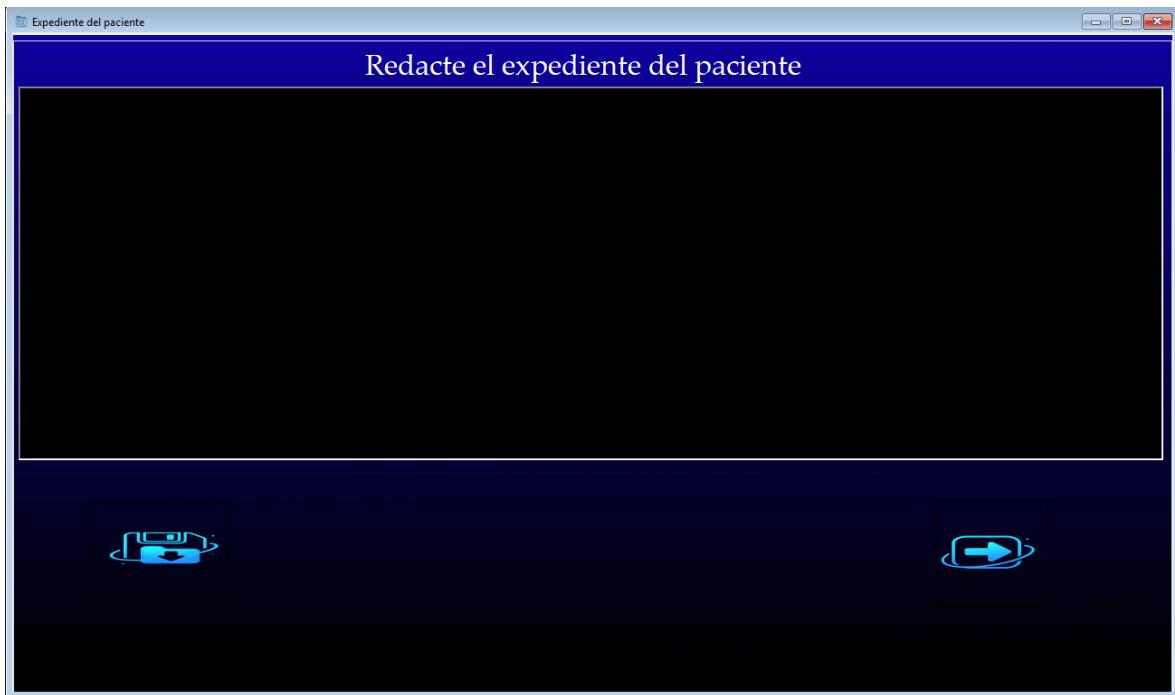
                MessageBox.Show("Expediente guardado correctamente");

                expedienteExiste = true; // Si era nuevo, ya existe
ahora.
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error al guardar el expediente: " +
ex.Message);
            }
        }

        private void btnCancelar_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}

```

## Diseño



## Expediente

La lógica del expediente permite que si no existe uno se pueda visualizar el richtextbox de manera vacía para poder redactar uno, requerimos al igual que para la cita un id de paciente seleccionado y adicionalmente requerimos una variable de tipo booleana inicializada en false lo que permitirá determinar si existe o no nuestro expediente y así mismo creamos un evento de acción público de expediente agregado. Por lo que primero traemos el id seleccionado a través del constructor y posteriormente vamos a crear un método para cargar el expediente, este método utiliza una propiedad de sql llamada data reader a través de un select para realizar una consulta a la base de datos instanciamos un reader y utilizando el id seleccionado y un comando de sql vamos a leer el dato guardado en la celda a través del reader y si este contiene un texto entonces actualizamos la variable booleana como true y en el caso contrario de no existir inicializamos la variable como false y dejamos el textbox vacío, este método se ejecuta en un evento de carga del formulario.

## Confirmar expediente

Ahora volvemos a entrar a la base de datos y creamos un string llamado query con nuestra variable booleana si el expediente existe vamos a ejecutar un update a través del query para que al guardar lo nuevamente redactado esto no afecte a los valores anteriores y si no existe entonces se ejecuta un insert para agregar nuevos caracteres de texto, ejecutamos el cmd con los valores correspondientes del query y si el formulario hizo un insert, es decir si guardo un nuevo valor que antes no existía entonces actualizamos la variable booleana a true.

## Modificar pacientes

### Código completo

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
//using Microsoft.Data.SqlClient;
using MySql.Data.MySqlClient;
namespace Astra
{
    public partial class Form7 : Form
    {
        string ruta;
        string cadenaConexion;
        private int idpacienteseleccionado;
        string nombre;
        string apellido;
        public event Action PacienteActualizado;
        public Form7(int IdPaciente)
        {
            InitializeComponent();
            idpacienteseleccionado = IdPaciente;
            // ruta = Path.Combine(Application.StartupPath,
@"Data\AstraDB.mdf");
            cadenaConexion =
"Server=localhost;Database=Astra;Uid=root;Pwd=277353;";

        }
        private bool arrastrar = false;
        private Point puntoInicio;
        private void panel2_MouseDown(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
            {
                arrastrar = true;
                puntoInicio = new Point(e.X, e.Y);
            }
        }
        private void panel2_MouseMove(object sender, MouseEventArgs e)
        {
            if (arrastrar)
            {
                Point p = PointToScreen(e.Location);
                this.Location = new Point(p.X - puntoInicio.X, p.Y -
puntoInicio.Y);
            }
        }
        private void panel2_MouseUp(object sender, MouseEventArgs e)
        {
            arrastrar = false;
        }
    }
}
```

```

        }

        private void CargarDatosAnteriores()
        {
            string edadprevia = "";
            string pesoprevio = "";
            string alturaprevia = "";
            string padecimiento_previo = " ";
            string alergiaprevia = "";

            using(MySqlConnection con = new MySqlConnection(cadena_conexion))
            {
                try
                {

                    con.Open();

                    string consulta = "SELECT Edad, Altura, Peso FROM
Pacientes WHERE IdPaciente = @IdPaciente";
                    using (MySqlCommand cmd = new MySqlCommand(consulta,
con))
                    {
                        cmd.Parameters.AddWithValue("@IdPaciente",
idpacienteseleccionado);
                        MySqlDataReader reader = cmd.ExecuteReader();

                        if (reader.Read())
                        {
                            edadprevia= reader["Edad"].ToString();
                            alturaprevia = reader["Altura"].ToString();
                            pesoprevio = reader["Peso"].ToString();

                            txtEdadPrevio.Text = edadprevia;
                            txtAlturaPrevia.Text = alturaprevia;
                            txtPesoPrevio.Text = pesoprevio;

                        }
                        reader.Close();
                    }
                }
            }

            string consulta2 = "SELECT Alergia FROM Alergias WHERE
IdPaciente = @IdPaciente ";
            using(MySqlCommand cmd2 = new MySqlCommand(consulta2,
con))
            {
                cmd2.Parameters.AddWithValue("@IdPaciente",
idpacienteseleccionado);
                MySqlDataReader reader2 = cmd2.ExecuteReader();

                if (reader2.Read())
                {

```

```
        alergiaprevia = reader2["Alergia"].ToString();

        txtAlergiasPrevio.Text = alergiaprevia;

    }

    reader2.Close();

}

string consulta3 = "SELECT Padecimiento FROM
Padecimientos WHERE IdPaciente = @IdPaciente";
using(MySqlCommand cmd3 = new MySqlCommand(consulta3,
con))
{
    cmd3.Parameters.AddWithValue("@IdPaciente",
idpacienteseleccionado);
    MySqlDataReader reader3 = cmd3.ExecuteReader();

    if (reader3.Read())
    {

        padecimientoprevio =
reader3["Padecimiento"].ToString();
        txtPadecimientosPrevios.Text =
padecimientoprevio;

    }
    reader3.Close();
}

}

catch(Exception ex)
{
    MessageBox.Show("Error al cargar datos previos " +
ex.Message);
    return;
}

}
```

```

        }
    private void CargarNombrePaciente()
    {
        using(MySqlConnection con = new MySqlConnection(cadena_conexion))
        {
            try
            {
                con.Open();
                string consulta = "SELECT Nombre, Apellido FROM Pacientes
WHERE IdPaciente = @IdPaciente";
                MySqlCommand cmd = new MySqlCommand(consulta, con);
                cmd.Parameters.AddWithValue("@IdPaciente",
idpacienteseleccionado);
                //Lector sql que permite leer los datos del paciente
                nombre y apellido
                MySqlDataReader reader = cmd.ExecuteReader();
                //Si el lector lee los datos, los asigna a las variables
                y las muestra en el textbox
                if (reader.Read())
                { //Si el lector lee los datos, los asigna a las
                variables y las muestra en el textbox
                    nombre = reader["Nombre"].ToString();
                    apellido = reader["Apellido"].ToString();
                    //Concatena el nombre y apellido
                    txtPaciente.Text = $"{nombre} + {apellido}";
                }
                else
                {
                    txtPaciente.Text = "Paciente no encontrado";
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error al cargar el nombre del paciente:
" + ex.Message);
            }
        }
    }
    private void Form7_Load(object sender, EventArgs e)
    {
        CargarNombrePaciente();
        CargarDatosAnteriores();
    }
    private void btnConfirmar_Click(object sender, EventArgs e)
    {

        int edad;
        double peso;
        double altura;
        string alergia;
        string padecimiento;
        try
        {
            edad = int.Parse(txtEdad.Text);
            peso = double.Parse(txtPeso.Text);
            altura = double.Parse(txtAltura.Text);
            alergia = txtAlergias.Text;
        }
    }
}

```

```

        padecimiento = txtPadecimientos.Text;

        if( edad<=0 || edad>= 100)
        {

            MessageBox.Show("Ingrese una edad valida ",
"Advertencia", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return;
        }

    }
    catch (Exception)
    {
        MessageBox.Show("Por favor, ingrese valores numéricos válidos
para Edad, Peso y Altura.", "Error de entrada", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        return;
    }

    using (MySqlConnection con = new
MySqlConnection(cadena_conexion))
{
    try
    {
        con.Open();
        string actualizar = "UPDATE Pacientes SET Edad = @Edad,
Altura = @Altura, Peso = @Peso";

        using(MySqlCommand cmd = new MySqlCommand(actualizar,
con))
        {
            cmd.Parameters.AddWithValue("@Edad", edad);
            cmd.Parameters.AddWithValue("@Peso", peso);
            cmd.Parameters.AddWithValue("Altura", altura);
            cmd.ExecuteNonQuery();
        }

        string actualizaralergia = "UPDATE Alergias SET Alergia =
@Alergia WHERE IdPaciente = @IdPaciente";
        using(MySqlCommand cmd2 = new
MySqlCommand(actualizaralergia, con))
        {

            cmd2.Parameters.AddWithValue("@IdPaciente",
idpaciente seleccionado);
            cmd2.Parameters.AddWithValue("@Alergia", alergia);
            cmd2.ExecuteNonQuery();

        }

        string actualizarPadecimiento = "UPDATE Padecimientos SET
Padecimiento = @Padecimiento WHERE IdPaciente = @IdPaciente";
        using(MySqlCommand cmd3 = new
MySqlCommand(actualizarPadecimiento, con))
        {

            cmd3.Parameters.AddWithValue("@IdPaciente",
idpaciente seleccionado);
            cmd3.Parameters.AddWithValue("@Padecimiento",
padecimiento);
        }
    }
}

```

```

        }

        MessageBox.Show("Datos actualizados del paciente ");
        PacienteActualizado.Invoke();
        this.Close();

    }
    catch (Exception ex)
    {
        MessageBox.Show("Error al actualizar los datos del
paciente: " + ex.Message);
    }
}

}

private void btnCerrar_Click(object sender, EventArgs e)
{
    this.Hide();
}
}

```

Este formulario permite a través de un select visualizar la información previa de los pacientes y posteriormente permite realizar modificaciones a esta misma actualizándolo en la base de datos a través de un update a los valores pertenecientes a un ID de paciente específico.

## Base de datos

### MySQL

Es importante importar la librería de MySql.Data.SqlClient;

De manera que nuestra aplicación funciona de la siguiente manera, nosotros tenemos una cadena de conexión que hace una conexión a la base de datos sql a través del puerto 3306 y un usuario y contraseña anteriormente definidas, posteriormente el script busca generar las tablas en el caso de que no existan garantizando que la aplicación funcione correctamente sin una intervención manual del usuario, mientras la conexión este abierta vamos a poder interactuar a través de nuestros diferentes menus que ejecutan dentro del propio código scripts de MySql sin que el usuario tenga que escribirlos en la base de datos, estos mismos solo reciben valores a través de la interfaz grafica y los eventos de los diferentes botones.

```

cadenaConexion = "Server=localhost;Database=Astra;Uid=root;Pwd=277353;" ;

create database Astra;
Drop database Astra;
USE Astra;
    -- archivo: astra_schema.sql
    CREATE DATABASE IF NOT EXISTS `Astra`
        DEFAULT CHARACTER SET utf8mb4
        DEFAULT COLLATE utf8mb4_general_ci;
    USE `Astra`;

    -- Tabla Pacientes (creada primero porque las demás referencian su
PK)
    CREATE TABLE IF NOT EXISTS `Pacientes` (
        `IdPaciente` INT NOT NULL AUTO_INCREMENT,
        `Nombre` VARCHAR(100) NOT NULL,
        `Apellido` VARCHAR(100) NOT NULL,
        `Edad` INT NULL,
        `Altura` FLOAT NULL,
        `Peso` FLOAT NULL,
        `Proxima_cita_Fecha` DATETIME NULL,
        `Hora` TIME NULL,
        PRIMARY KEY (`IdPaciente`)
    ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

    -- Tabla Doctores (Id AUTO_INCREMENT + FK a Pacientes, IdPaciente
NULLABLE)
    CREATE TABLE IF NOT EXISTS `Doctores` (
        `Id` INT NOT NULL AUTO_INCREMENT,
        `Usuario` VARCHAR(100) NOT NULL,
        `Contraseña` VARCHAR(255) NOT NULL, -- mantenido tal como en tu
código; se recomienda usar hash
        `Doctor` VARCHAR(50) NULL,
        `IdPaciente` INT NULL,
        PRIMARY KEY (`Id`),
        UNIQUE KEY `UK_Doctor_Usuario` (`Usuario`),
        CONSTRAINT `FK_Doctores_Pacientes` FOREIGN KEY (`IdPaciente`)
            REFERENCES `Pacientes` (`IdPaciente`)
            ON DELETE SET NULL
            ON UPDATE CASCADE
    ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

    -- Tabla Alergias
    CREATE TABLE IF NOT EXISTS `Alergias` (

```

```

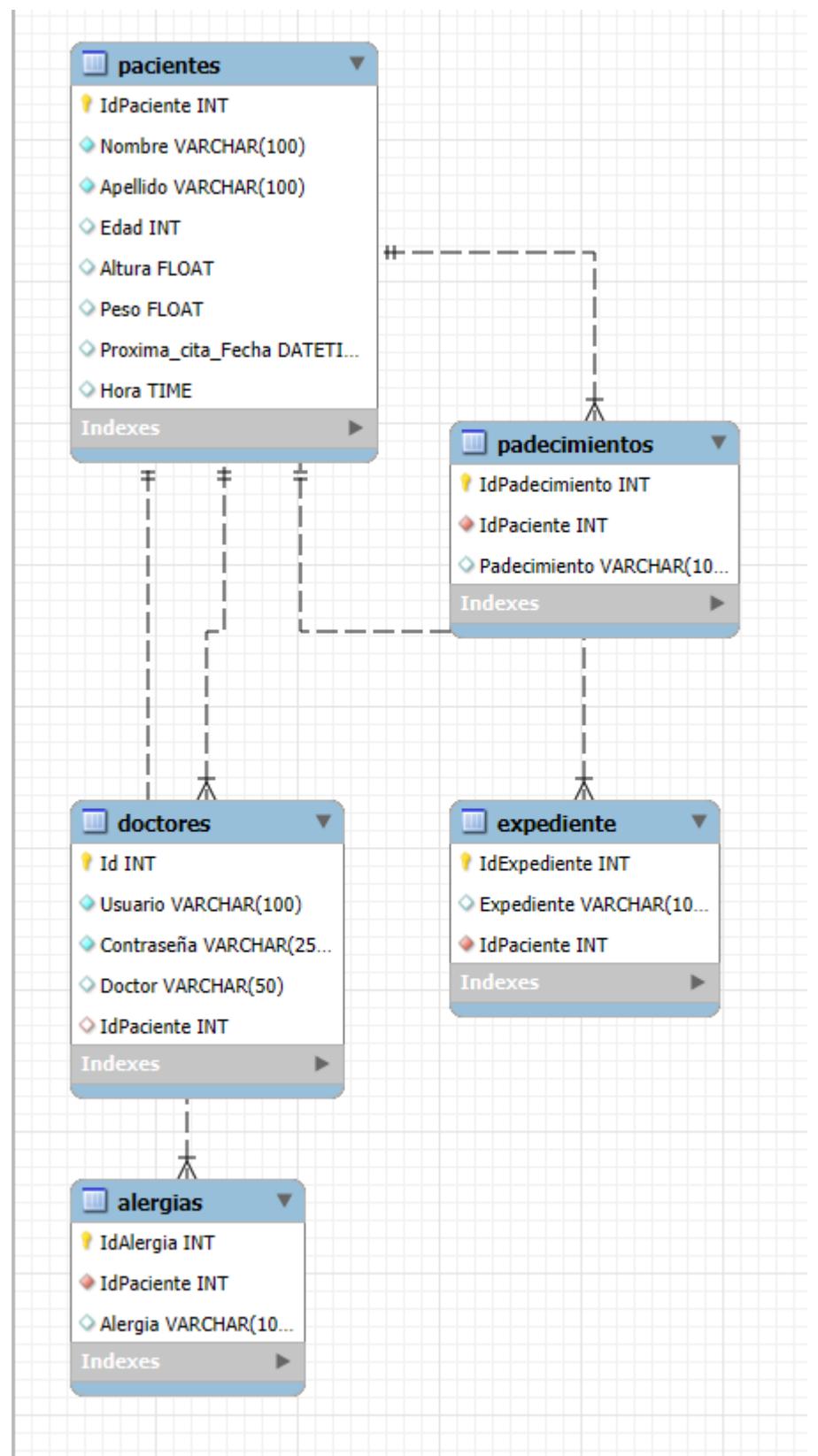
`IdAlergia` INT NOT NULL AUTO_INCREMENT,
`IdPaciente` INT NOT NULL,
`Alergia` VARCHAR(100) NULL,
PRIMARY KEY(`IdAlergia`),
KEY `idx_alergias_paciente`(`IdPaciente`),
CONSTRAINT `FK_Alergias_Pacientes` FOREIGN KEY(`IdPaciente`)
REFERENCES `Pacientes`(`IdPaciente`)
ON DELETE CASCADE
ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Tabla Padecimientos
CREATE TABLE IF NOT EXISTS `Padecimientos` (
`IdPadecimiento` INT NOT NULL AUTO_INCREMENT,
`IdPaciente` INT NOT NULL,
`Padecimiento` VARCHAR(100) NULL,
PRIMARY KEY(`IdPadecimiento`),
KEY `idx_padecimientos_paciente`(`IdPaciente`),
CONSTRAINT `FK_Padecimientos_Pacientes` FOREIGN KEY
(`IdPaciente`)
REFERENCES `Pacientes`(`IdPaciente`)
ON DELETE CASCADE
ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Tabla Expediente
CREATE TABLE IF NOT EXISTS `Expediente` (
`IdExpediente` INT NOT NULL AUTO_INCREMENT,
`Expediente` VARCHAR(100) NULL,
`IdPaciente` INT NOT NULL,
PRIMARY KEY(`IdExpediente`),
KEY `idx_expediente_paciente`(`IdPaciente`),
CONSTRAINT `FK_Expediente_Pacientes` FOREIGN KEY
(`IdPaciente`)
REFERENCES `Pacientes`(`IdPaciente`)
ON DELETE CASCADE
ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

```

## Tablas



La primera tabla que modelamos para la base de datos es la de Pacientes ya que esta pasa su IdPaciente a las demás a través de claves foráneas esta nos permite guardar en variables los diferentes datos acerca de un paciente según lo requiera el doctor.

Después se modela la tabla que controla la gestión de usuarios de la aplicación: La tabla doctores, cada Paciente tiene uno o varios doctores según los usuarios que se registren en la aplicación y de esta manera determinamos que solo se permita un usuario activo a la vez-

Posteriormente para la normalización modelamos tablas de padecimiento y alergias que a través de la aplicación nos guardan en listas cada alergia o padecimiento registrado de manera que no haya datos repetidos en una sola fila.

La tabla expediente nos permite guardar lo redactado por el doctor en turno acerca de la consulta y enlaza a través de los ID a que paciente pertenece cada expediente.