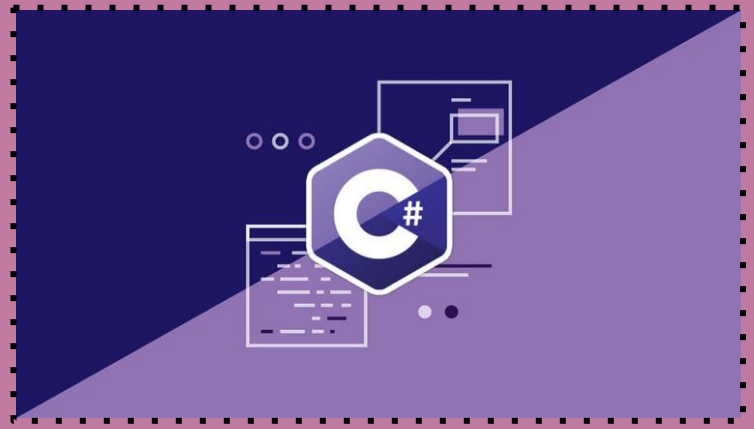
Presentación 1er. Proyecto: Lightning Talk

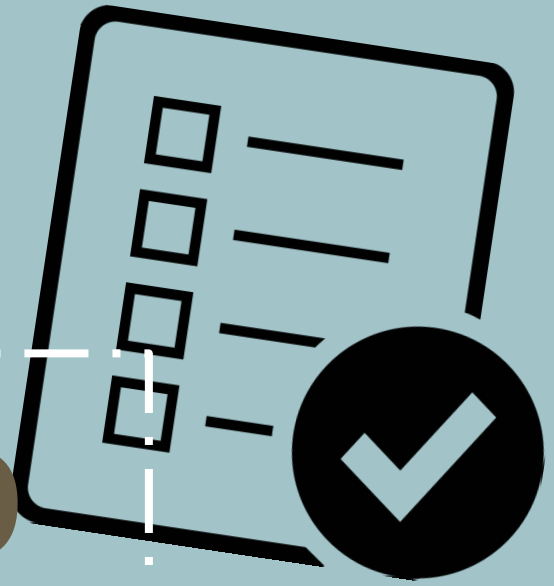
Sara Castro

Luz Rodríguez

SisLey



OBJETIVO



Anállsls

¿Cómo puedo facilitar el acceso de Leyes a los interesados en una forma óptima?

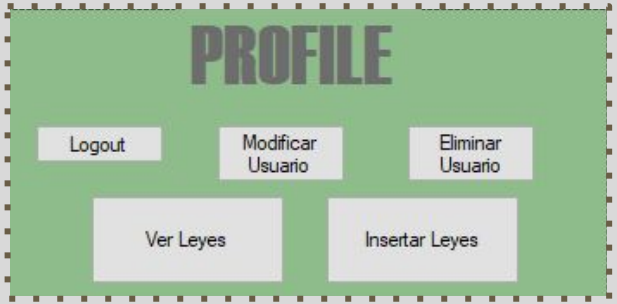
¿Por qué C# es la mejor opción?

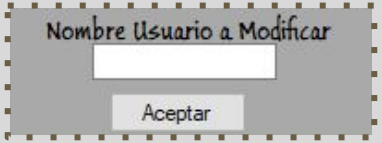
¿Qué entradas y salidas necesito para la creación del usuario?

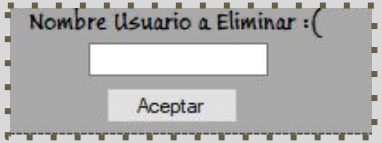
¿Cúal fue el procedimiento?

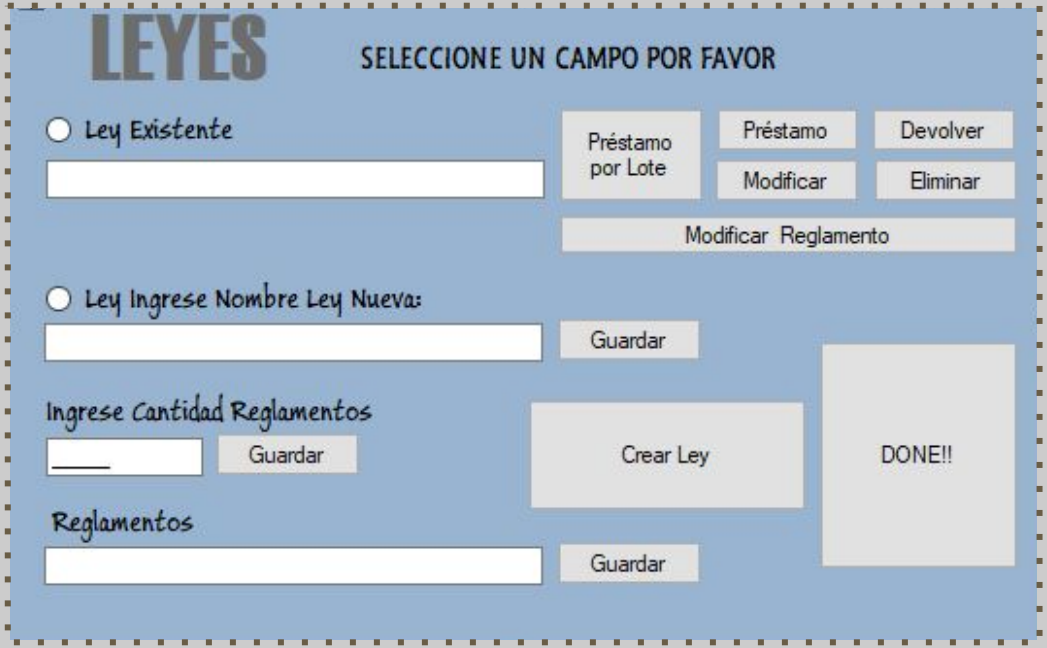
¿Cuáles fueron las restricciones?

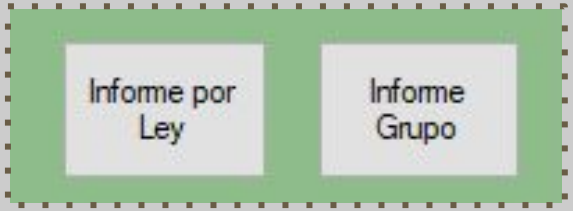
Utilización de Paneles para la facilitación de la Implementación





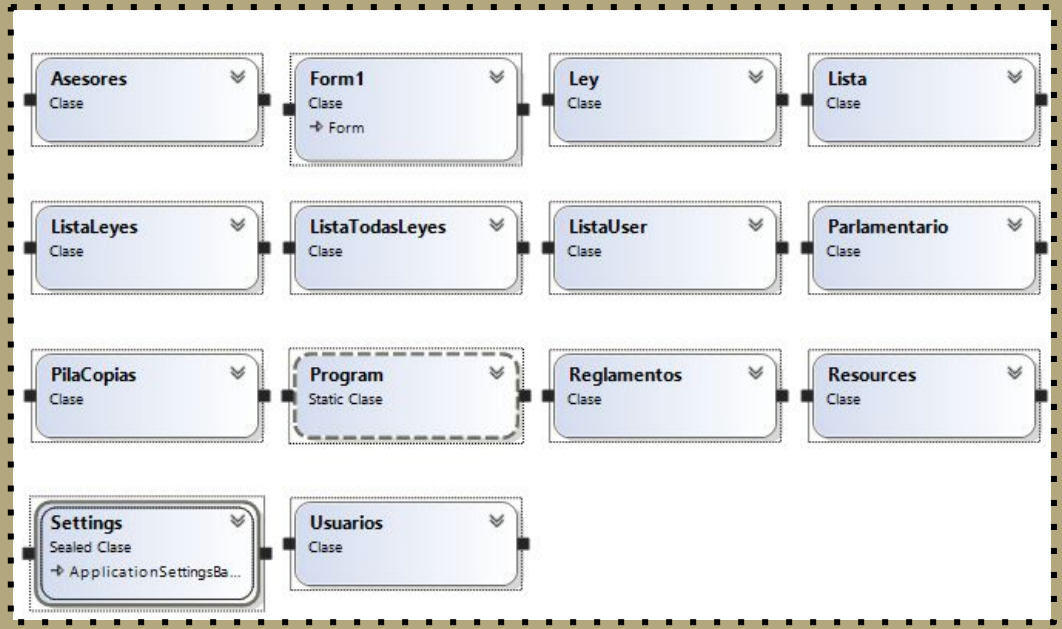



LEYES
SELECCIONE UN CAMPO POR FAVOR
Ley Existente
Préstamo por Lote
Préstamo
Devolver
Modificar
Eliminar
Modificar Reglamento
Ley Ingrese Nombre Ley Nueva:
Guardar
Ingrese Cantidad Reglamentos
____
Guardar
Crear Ley
DONE!!
Reglamentos
Guardar

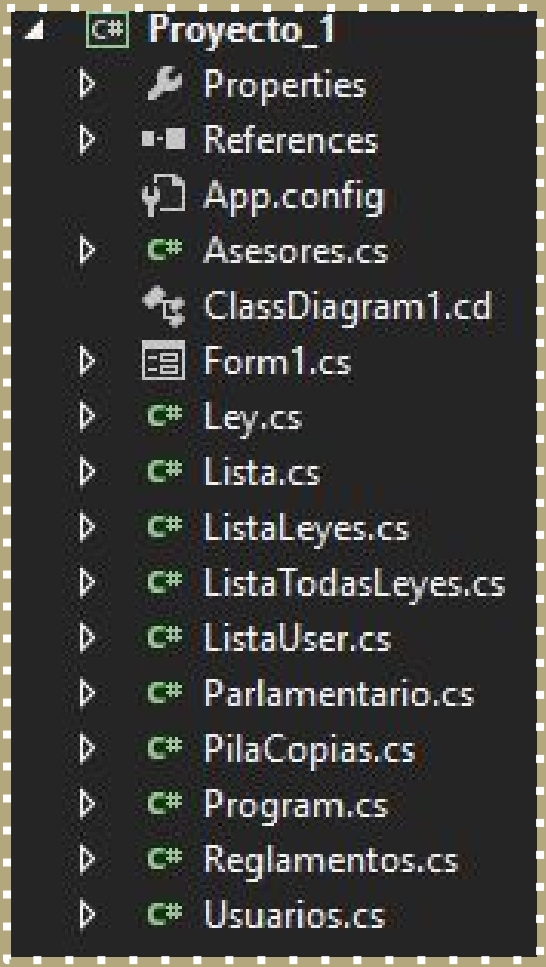

Informe por Ley
Informe Grupo


Ingrese el Nombre de la Ley a Devolver
Devolver

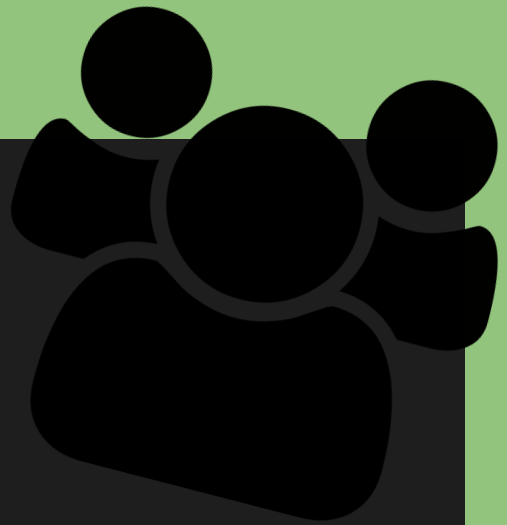
CÓDIGO


Asesores
Clase
Form1
Clase
Form
Ley
Clase
Lista
Clase
ListaLeyes
Clase
ListaTodasLeyes
Clase
ListaUser
Clase
Parlamentario
Clase
PilaCopias
Clase
Program
Static Clase
Reglamentos
Clase
Resources
Clase
Settings
Sealed Clase
ApplicationSettingsBa...
Usuarios
Clase

Clases


Proyecto_1
Properties
References
App.config
Asesores.cs
ClassDiagram1.cd
Form1.cs
Ley.cs
Lista.cs
ListaLeyes.cs
ListaTodasLeyes.cs
ListaUser.cs
Parlamentario.cs
PilaCopias.cs
Program.cs
Reglamentos.cs
Usuarios.cs

USUARIOS



```
public class Usuarios
{

    public Lista ListaAsesorDentroUsuario;
    public string nombre;
    public string [] nombreAse;
    private string password;
    public ListaLeyes listadoLeyes;
    int asesoresc;
    /// <summary>
    /// Clase Usuarios que posee un metodo constructor "Usuarios"
    /// </summary>
    /// <param name="nombre"></param>
    /// <param name="asesoresc"></param>
    /// <param name="nombreAse"></param>
    /// <param name="contra"></param>
    /// <param name="lis"></param>
    public Usuarios (string nombre, int asesoresc, Lista nombreAse,  string contra, ListaLeyes lis){
        this.nombre = nombre;
        this.asesoresc = asesoresc;
        this.ListaAsesorDentroUsuario = nombreAse;
        this.password = contra;
        this.listadoLeyes = lis;
}
```

CREACIÓN DE LOS USUARIOS

```
private void btnCreado_Click(object sender, EventArgs e)
{
        string nom;
        int numAse;
        nom = txtNombreDip.Text;
        string contraseña;
        contraseña = txtContraseña.Text;
        numAse = int.Parse(mTxtCasesores.Text);
        us = new Usuarios(nom, numAse, listaAsesor, contraseña, listaDeLeyes);
```

ELIMINACIÓN DE LOS USUARIOS

```
string UserR;
UserR = txtNomEliminar.Text;
for (int k = 0; k < listaUsuarios.cima; k++)
{
    if (UserR == listaUsuarios.Usuarios[k].nombre)
    {
        listaUsuarios.Remove(k);
        MessageBox.Show("Eliminado");
        pnElimina.Visible = false;
        pnInicio.Visible = true;
    }
    if (k == listaUsuarios.limite - 1)
    {
        MessageBox.Show("No Existe");
    }
```

```
string nomModificar;
nomModificar = txtUsuarioModificar.Text;

for (int k = 0; k < listaUsuarios.cima; k++)
{
    if (nomModificar == listaUsuarios.Usuarios[k].nombre)
    {
        pnCreate.Visible = true;
        listaUsuarios.Remove(k);
        variableSesion = -1;
        pnModificar.Visible = false;

    }
    if (k == listaUsuarios.limite - 1)
    {
        MessageBox.Show("No Existe");
    }
```

MODIFICACIÓN DE LOS USUARIOS

```
public class ListaTodasLeyes
{

    public Ley[] Leyes;
    public int cima2 = 0;
    public int limite2;
    private bool vacio2;
    private bool lleno2;
    private Ley valor3;


     public ListaTodasLeyes()
    {
        Leyes= new Ley[200];
        limite2 = 200;
    }
    private bool Valida_vacia_Leyes()
    {
        if (Leyes[0] == default(Ley))
        {
            vacio2 = true;
        }
        else
        {
            vacio2 = false;
        }
        return vacio2;
    }
```

```
private bool Valida_lleno_Leyes()
{
    if (Leyes[0] != default(Ley))
    {
        lleno2 = true;
    }
    else
    {
        lleno2 = false;
    }
    return lleno2;
}

public void Add_Leyes(Ley d)
{

    if (cima2 < limite2)
    {
        Leyes[cima2] = d;
        cima2++;

    }

}
```

```
public Ley Remove(int puntero)
{
    if (Valida_lleno_Leyes() && puntero <= cima2)
    {
        if (puntero == cima2)
        {
            valor3 = Leyes[puntero];
            Leyes[cima2] = default(Ley);
            cima2--;
        }
        else
        {
            valor3 = Leyes[puntero];
            for (int i = puntero; i < cima2; i++)
            {
                Leyes[i] = Leyes[i + 1];
            }
            Leyes[cima2] = default(Ley);
            cima2--;
        }
    }
    else
    {
        Console.WriteLine("La lista no posee valor en la posicion especificada");
    }
    return valor3;
}
```

Lista Leyes

PILA COPIAS

```
public class PilaCopias
 {
      private int cima;
     private int limite;
     private string valor;
     private string[] arreglo;
     private bool vacio;
     private bool lleno;

     /// <summary>
     /// Constructor de la clase Cola
     /// </summary>
     /// <param name="Ilimite">El limite de la cola</param>
     public PilaCopias(int Ilimite)
     {
         cima = -1;
         limite = Ilimite;
         valor = "";
         arreglo = new string[limite];
         vacio = false;
         lleno = false;
     }
```

```
    private bool Valida_vacia()
    {
        if (arreglo[0] == default(string))
        {
            vacio = true;
        }
        else
        {
            vacio = false;
        }
        return vacio;
    }

    /// <summary>
    /// Valida si la pila esta lleno
    /// </summary>
    /// <returns>Retorna un valor verdadero si esta llena y falso si esta vacia</returns>
    private bool Valida_lleno()
    {
        if (arreglo[0] != default(string))
        {
            lleno = true;
        }
        else
        {
            lleno = false;
        }
        return lleno;
    }
```

```
    public void Push(string valor_ingresado)
    {
        if (Valida_vacia() && cima < 0)
        {
            arreglo[0] = valor_ingresado;
            cima++;
        }
        else
        {
            if (cima < limite)
            {
                cima++;
                arreglo[cima] = valor_ingresado;
            }
        }
    }
```

```
    /// <summary>
    /// La funcion devuelve el primer dato que ingreso
    /// </summary>
    /// <returns>El primer dato que entro es el primero que sale</returns
    public string Pop()
    {
        if (Valida_lleno())
        {
            valor = arreglo[cima];
            arreglo[cima] = default(string);
            cima--;
        }
        else
        {
            valor = "Entro a la Lista de Espera";
        }
        return valor;
    }
```

LEYES

```
public class Ley
{
    public Lista listaReglamentos;
    public string nomLey;
    private int cantReglamentos;
    public PilaCopias pilaCopies = new PilaCopias(5);


    /// <summary>
    /// Clase Ley que contiene las copias, el nombre y la lista de reglamentos
    /// </summary>
    /// <param name="nL"></param>
    /// <param name="cRegla"></param>
    /// <param name="nomRegla"></param>
    public Ley(string nL, int cRegla, Lista nomRegla)
    {
        this.nomLey = nL;
        this.listaReglamentos = nomRegla;
        this.cantReglamentos = cRegla;
        pilaCopies.Push(nL);
        pilaCopies.Push(nL);
        pilaCopies.Push(nL);
        pilaCopies.Push(nL);
    }

}
```

CREACIÓN DE LEYES

```
l = new Ley(nomLey, d, listaReglamentos);
if (variableSesion != -1 && contadorAsesor == -1)
{
    listaUsuarios.Usuarios[variableSesion].AgregaLeyes(l);
}
if (contadorAsesor != -1)
{
    listaUsuarios.Usuarios[variableSesion].ListaAsesorDentroUsuario.listaAsesor[contadorAsesor].AgregaLeyes(l);
}
```

ELIMINACIÓN DE LEYES

```
{
    string LeyEliminada;
    LeyEliminada = txtLeyExistente.Text;
    for (int k = 0; k < listaDeLeyesCompleta.cima2; k++)
    {
        if (LeyEliminada == listaDeLeyesCompleta.Leyes[k].nomLey)
        {
            listaDeLeyesCompleta.Remove(k);
            MessageBox.Show("Eliminado");
            pnLeyes.Visible = false;
            pnUsuario.Visible = true;
            btnLogout.Visible = true;
```

MODIFICACIÓN DE LEYES

```
private void btnModLey_Click(object sender, EventArgs e)
{
    string nomModificarLey;
    nomModificarLey = txtLeyExistente.Text;
    for (int k = 0; k < listaDeLeyesCompleta.cima2; k++)
    {
        if (nomModificarLey == listaDeLeyesCompleta.Leyes[k].nomLey)
        {
            if (variableSesion != -1)
            {
                listaDeLeyesCompleta.Remove(k);
                listaUsuarios.Usuarios[variableSesion].listadoLeyes.Remove(k);
```

```
rbtnLeyNueva.Checked = true;
lblLeyMod.Visible = true;
txtLeyNueva.Clear();
txtLeyNueva.Visible = true;
btnGuardarLey.Visible = true;
lblCantidadReglamentos.Visible = true;
mTxtCantRegla.Clear();
mTxtCantRegla.Visible = true;
btnGuardarCantRegla.Visible = true;
lblReglamento.Visible = true;
txtReglamento.Visible = true;
txtReglamento.Clear();
btnGuardarRegla.Visible = true;
btnLeyCreada.Visible = true;
rLblLeyExistente.Visible = false;
btnAgregarLeyExistente.Visible = false;
btnprest.Visible = false;
button4.Visible = false;
btnModLey.Visible = false;
btnEliminarLey.Visible = false;
txtLeyExistente.Visible = false;
txtLeyExistente.Clear();
btnFunciones.Visible = true;
```