Introducción

En el proyecto presentado, se aborda la problemática de la validación de cadenas de texto según reglas específicas de gramáticas utilizando expresiones regulares en C#. Se trata de una tarea común en la programación, donde es necesario asegurarse de que los datos ingresados por los usuarios cumplan con requisitos predefinidos. En este caso, se exploran cuatro ejemplos de validación de cadenas que incluyen contraseñas seguras, números de teléfono, números de Seguro Social (SSN) y direcciones físicas.

Metodología

La metodología empleada para resolver esta problemática se basa en el uso de expresiones regulares y el lenguaje de programación C#. A continuación, se describe el método para validar las cadenas de texto mediante expresiones regulares:

Expresión Regular para Contraseñas Seguras: La expresión regular utilizada en este caso busca patrones que cumplan con los siguientes requisitos: al menos 8 caracteres, al menos una letra mayúscula, una letra minúscula y un número. Se utiliza la función Regex.IsMatch para verificar si una cadena cumple con este patrón.

```
1 referencia
private void btnContraseña_Click(object sender, EventArgs e)
{
    string input = TextContraseña.Text;
    string patron = @"^(?=.*[a-z])(?=.*[A-Z])(?=.*\d).{8,}$";

    if (Regex.IsMatch(input, patron))
    {
        MessageBox.Show("Cadena SI aceptada");
    }
    else
    {
        MessageBox.Show("Cadena NO aceptada");
    }
}
```

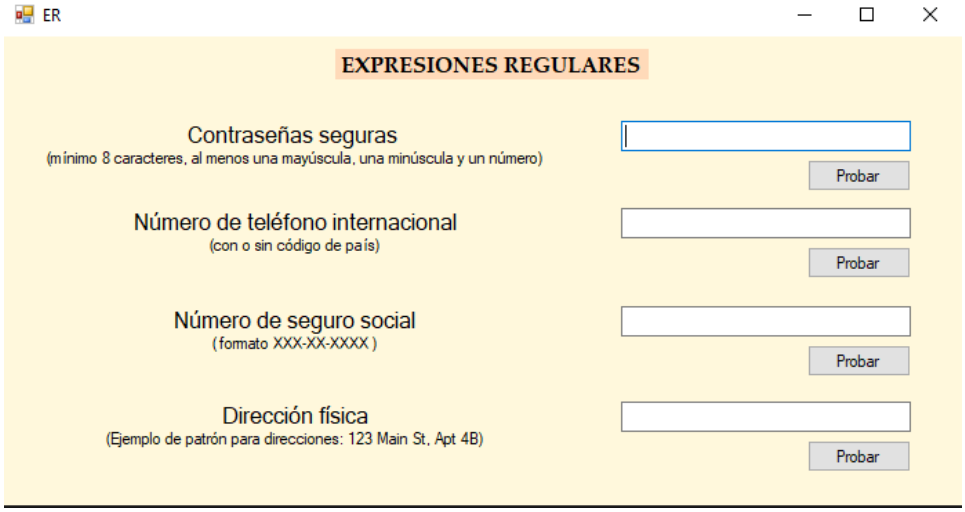
Expresión Regular para Números de Teléfono: Esta expresión regular valida números de teléfono en formato internacional, permitiendo diferentes formatos, incluidos prefijos internacionales opcionales y paréntesis en el código de área. La función Regex.IsMatch se usa para validar la entrada del usuario.

```
1 referencia
private void btnTelefono_Click(object sender, EventArgs e)
{
    string input = TextTelefono.Text;
    string patron = @"^\+?\d{1,3}[-.\s]?\(?\d{1,4}\)?[-.\s]?\d{1,10}$";

    if (Regex.IsMatch(input, patron))
    {
        MessageBox.Show("Cadena SI aceptada");
    }
    else
    {
        MessageBox.Show("Cadena NO aceptada");
    }
}
```

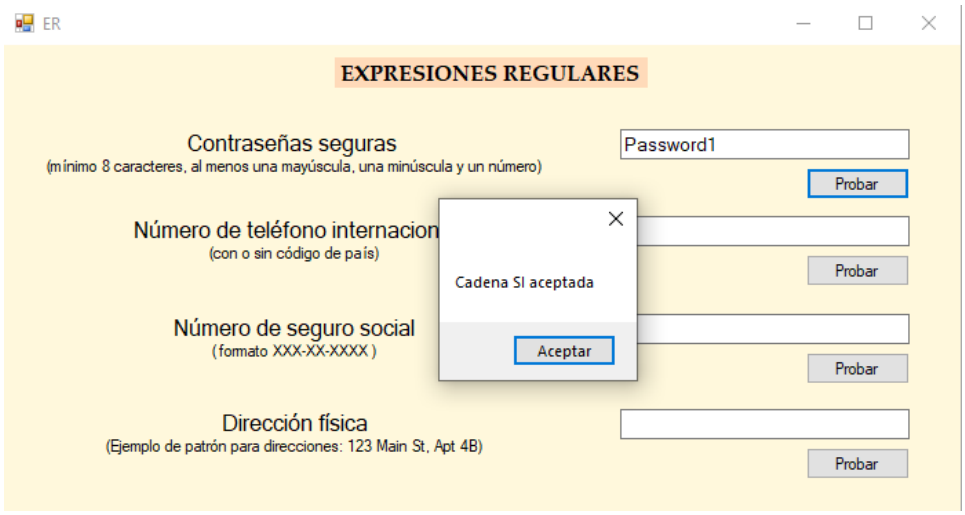
```
1 referencia
private void btnDireccion_Click(object sender, EventArgs e)
{
    string input = TextDireccion.Text;
    string patron = @"^\d+\s[A-Za-z]+\s(St|Rd|Ave|Blvd)\,\s(Ste|Apt)\s\d+[A-Za-z]*$";

    if (Regex.IsMatch(input, patron))
    {
        MessageBox.Show("Cadena SI aceptada");
    }
    else
    {
        MessageBox.Show("Cadena NO aceptada");
    }
}
```

Resultados

Ventana Principal



Probando cadena de contraseña SI aceptada

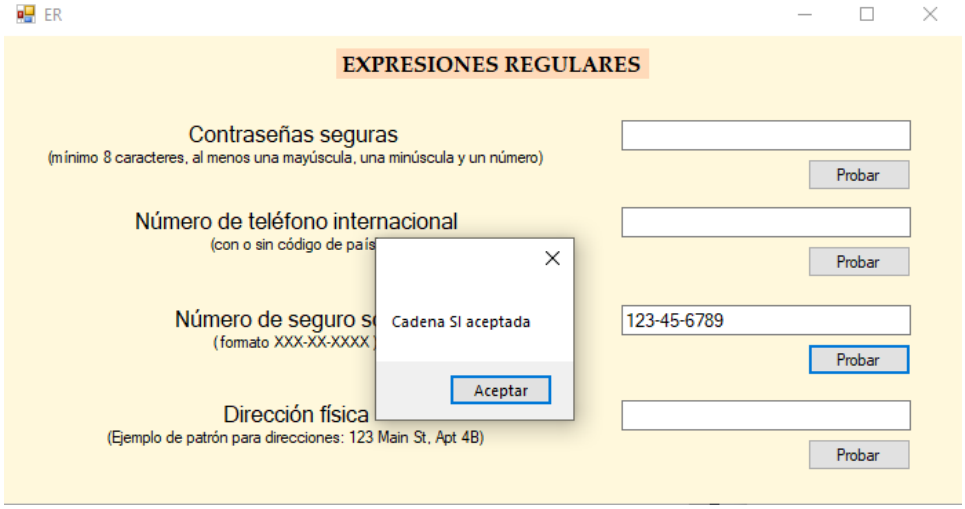


Probando cadena de contraseña NO aceptada

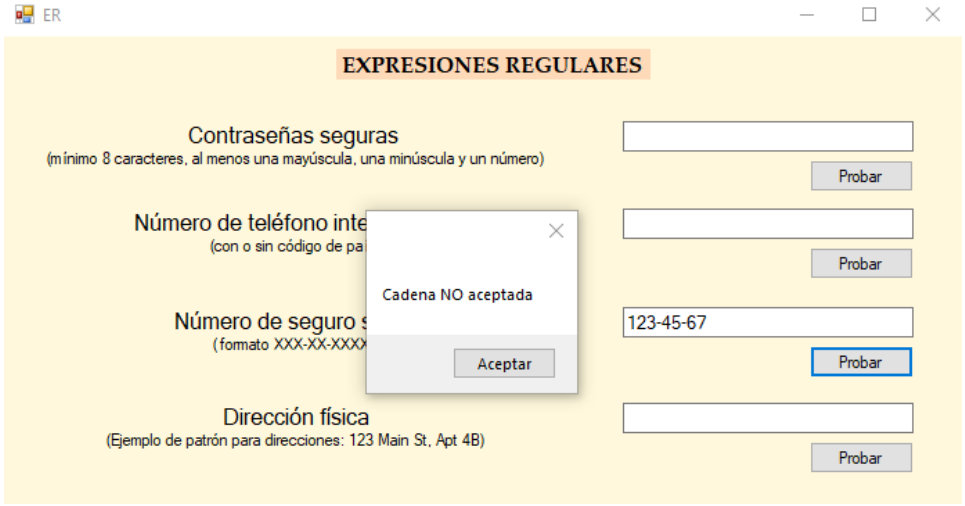
Probando cadena de telefono SI aceptada

Probando cadena de telefono NO aceptada

Probando cadena de Seguro Social SI aceptada



Probando cadena de Seguro Social NO aceptada



Probando cadena de Direccion SI aceptada

Probando cadena de Direccion NO aceptada

Conclusiones

La implementación de expresiones regulares en C# para validar cadenas de texto es una técnica efectiva y potente. Algunas conclusiones sobre el proyecto son las siguientes:

Las expresiones regulares permiten definir patrones de validación de manera concisa y precisa. Facilitan la verificación y el cumplimiento de reglas específicas de gramáticas. Sin embargo pueden volverse complicadas y difíciles de leer en patrones complejos. La validación se basa en la estructura de la cadena y no en la semántica real de los datos. Observaciones personales:

Es importante equilibrar la complejidad de las expresiones regulares para mantener la legibilidad del código. Las expresiones regulares son útiles para la validación, pero no siempre son la mejor opción para todos los escenarios de procesamiento de datos.