UNIVERSIDAD DE GUADALAJARA



CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

Inteligencia Artificial

Reporte de práctica

Nombre del alumno: Héctor Samuel Gómez Casillas Profesor: Erasmo Gabriel Martínez Soltero Título de la práctica: "Tarea 1. Expresiones Regulares"

Fecha: 04 septiembre 2023

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.

```
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");
    }
}
```

Expresión Regular para Números de Seguro Social (SSN): La expresión regular para SSN valida números en formato XXX-XXXXX. Se utiliza Regex.IsMatch para verificar si una cadena se ajusta a este formato.

```
referencia
private void btnSS_Click(object sender, EventArgs e)
{
    string input = TextSS.Text;
    string patron = @"^\d{3}-\d{2}-\d{4}$";

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

Expresión Regular para Direcciones Físicas: Esta expresión regular valida direcciones físicas en un formato específico que incluye números, nombres de calles y números de apartamentos. La función Regex.IsMatch se emplea para validar la entrada del usuario.

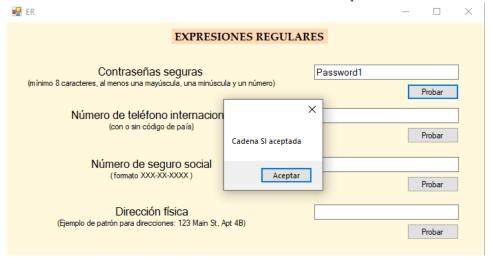
```
1 referencia
private void btnDireccion_Click(object sender, EventArgs e)
{
    string input = TextDireccion.Text;
    string patron = @"^\d+\s[A-Za-z]+\s(St|Rd|Ave|Blvd)\,\s(Ste|Apt)\s\d+[A-Za-z]*$";
    if (Regex.IsMatch(input, patron))
    {
        MessageBox.Show("Cadena SI aceptada");
    }
    else
    {
        MessageBox.Show("Cadena NO aceptada");
    }
}
```

Resultados

Ventana Principal



Probando cadena de contraseña SI aceptada



Probando cadena de contraseña NO aceptada



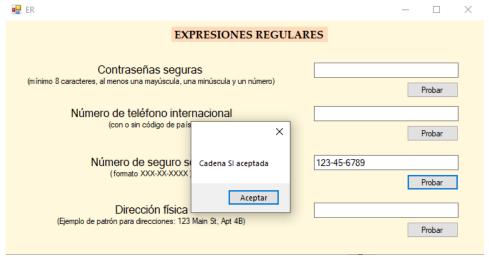
Probando cadena de telefono SI aceptada



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 🖷 ER EXPRESIONES REGULARES Contraseñas seguras (mínimo 8 caracteres, al menos una mavúscula, una minúscula y un número) Probar Número de teléfono internacional (con o sin código de país) Probar X Número de seguro so Cadena SI aceptada (formato XXX-XX-XXXXX) Probar Aceptar Dirección física 456 Elm Rd, Ste 100C (Ejemplo de patrón para direcciones: 123 Main St, Apt 4B) Probar Probando cadena de Direccion NO aceptada 🖷 ER EXPRESIONES REGULARES Contraseñas seguras (mínimo 8 caracteres, al menos una mayúscula, una minúscula y un número) Probar Número de teléfono internacional (con o sin código de p × Probar Número de seguro Cadena NO aceptada (formato XXX-XX-XX) Probar Aceptar Dirección físio Main St 156 (Ejemplo de patrón para direcciones: 123 Main St, Apt 4B)

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:

Probar

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.

Referencias

- NetMentor. (s. f.). Expresiones regulares en C. https://www.netmentor.es/entrada/expresiones-regulares. https://www.netmentor.es/entrada/expresiones-regulares
- Curso de C. (2015, 10 marzo). Usar expresiones regulares para extraer parte de un texto curso de C. https://csharp.com.es/usar-expresiones-regulares-para-extraer-parte-de-un-texto/