Simulador de IDE (Analizador Léxico)

Manual de Técnico

Versión: V1.0

Fecha: Septiembre, 2020

1. Tipo token reconocidos

Reconoce un lexema, identifica que tipo de lexema es, muestra en una tabla el tipo de token que encontró en el lexema, y al identificar uno de los siguientes tipos de token los pinta del color indicado en el richTxbox.

Tipo token: Entero	Color: Morado	Símbolo: 1 2 3 45
Tipo token: Decimal	Color: Celeste	Símbolo: -10 -9.2 1.1 0.9
Tipo token: Cadena	Color: Gris	Símbolo: “abcd” “ABCD”
Tipo token: Bolean	Color: Naranja	Símbolo: true false
Tipo token: Char	Color: Café	Símbolo: a A b B
Tipo token: Operadores aritméticos	Color: Azul oscuro	Símbolo: + - / * ++ --
Tipo token: Operadores relacionales	Color: Azul oscuro	Símbolo: < > <= >= == !=
Tipo token: Operadores lógicos	Color: Azul oscuro	Símbolo: || && !
Tipo token: Signos de agrupación	Color: Azul oscuro	Símbolo: ( )
Tipo token: Asignación y fin de sentencia	Color: Rosa	Símbolo: = ;
Tipo token: Palabras reservadas	Color: Verde	Símbolo: SI, SINO, SINO_SI, MENTRAS, HACER, DESDE, HASTA, INCREMENTO
Tipo token: Comentario	Color: Rojo	Símbolo: // /**/

1. Uso

RichTextBox:

Código:

```
bool richCambiado = false;
1 referencia
private void entradaRichTextBox_TextChanged(object sender, EventArgs e)
{
    richCambiado = true;

this.Text = string.Format("IDE - {0} *- ", manejador.getFileName());
    getLineaColumna();

}
```

```
2 referencias
public void getLineaColumna() {
    int linea = entradaRichTextBox.GetLineFromCharIndex(entradaRichTextBox.SelectionStart) + 1;
    int columna = entradaRichTextBox.SelectionStart - entradaRichTextBox.GetFirstCharIndexOfCurrentLine();
    getLineaLabel.Text = linea.ToString();
    getColumnaLabel.Text = columna.ToString();
}
```

Botón: compilar — Acción: analiza y pinta el código escrito por el usuario, y muestran en una tabla los tokens resultados del análisis, y si hay errores los muestra en otra tabla de errores.

```
Analizador analizador = new Analizador();
1 referencia
private void limpiarCodigoBoton_Click(object sender, EventArgs e)
{


    dataGridView1.DataSource = null;
    dataGridView2.DataSource = null;


    analizador.analizar(entradaRichTextBox, dataGridView1,dataGridView2);

    String direccionArchivo;
    direccionArchivo = manejador.getFileName();


        this.Text = string.Format("IDE - {0} - {1} tokens {2} errores", direccionArchivo, analizador.getNumeroToken(), analizador.getNumeroErrores());



}
```

Botón: guardar Acción: sobre escribe el archivo abierto con los cambies realizados por el usuario. Si el archivo en el que se está trabajando no está creado, abre una ventana para que el usuario seleccione la ruta donde desea guardarlo.

```
1 referencia
private void button1_Click_1(object sender, EventArgs e)
{
    richCambiado = false;
    manejador.guardar(manejador.getFileName(), entradaRichTextBox.Text);
    this.Text = string.Format("IDE - {0} - ", manejador.getFileName());
}
```

Botón: Archivo Acción: muestra las siguientes opciones: Nuevo, Guardar Como, Cargar

Botón: Nuevo Acción: Crea un archivo vacío, donde se limpian las tablas y todo queda como en la primera ejecución.

```
1 referencia
private void nuevoToolStripMenuItem_Click(object sender, EventArgs e)
{

    if (richCambiado == true)
    {
        if (MessageBox.Show("¿Quieres guardar los cambios realizados en tu codigo antes de crear un nuevo documento?", "IDE", MessageBoxButtons.YesNo) == DialogResult.Yes)
        {
            this.Text = string.Format("IDE - {0} - ", manejador.getFileName());
            manejador.guardarComo(saveFileDialog1, entradaRichTextBox.Text);
                richCambiado = false;


        }
        else
        {
            manejador.setFileName(null);
            entradaRichTextBox.Text = "";
            this.Text = string.Format("IDE - {0} - ", manejador.getFileName());
            richCambiado = false;
            dataGridView1.DataSource = null;
            dataGridView2.DataSource = null;



        }
    }
    manejador.setFileName(null);
    entradaRichTextBox.Text = "";
    this.Text = string.Format("IDE - {0} - ", manejador.getFileName());
    richCambiado = false;
    dataGridView1.DataSource = null;
    dataGridView2.DataSource = null;
}
```

Botón: Guardar Como

Acción: Abre una ventana donde el usuario debe seleccionar la ruta donde desea guardar el archivo que está editado, como también puede sobre escribir un ya existente.

Código:

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

    manejador.guardarComo(saveFileDialog1, entradaRichTextBox.Text);

    this.Text = string.Format("IDE - {0} - ", manejador.getFileName());
    richCambiado = false;
}
```

Botón: Cargar

Acción: Abre una ventana donde el usuario debe seleccionar la ruta donde se encuentra el archivo que desea editar.

Código:

```
1 referencia
private void cargarToolStripMenuItem_Click(object sender, EventArgs e)
{
    manejador.cargar(openFileDialog1, entradaRichTextBox);
    this.Text = string.Format("IDE - {0} - ", manejador.getFileName());
    richCambiado = false;

}
```

Botón: Salir

Acción: Cierra la aplicación, pero si el archivo es editado antes de presionar este botón, entonces le preguntara al usuario si desea guardar los cambios realizados o no.

```
1 referencia
private void IDE_FormClosing(object sender, FormClosingEventArgs e)
{

    if (richCambiado == true) {
        if (MessageBox.Show("¿Quieres guardar los cambios realizados en tu codigo antes de salir?", "IDE", MessageBoxButtons.YesNo) == DialogResult.No)
        {

            if (manejador.getFileName() == null)
            {

                manejador.guardarComo(saveFileDialog1, entradaRichTextBox.Text);
                richCambiado = false;
                this.Text = string.Format("IDE - {0} - ", manejador.getFileName());
            }
            else
            {
                richCambiado = false;
                manejador.guardar(manejador.getFileName(), entradaRichTextBox.Text);
                this.Text = string.Format("IDE - {0} - ", manejador.getFileName());
            }


        }
        else
        {
            Application.Exit();
        }
    }
    Application.Exit();


}
```

Botón: Exportar Error

Acción: toma todos los datos de la tabla de errores y los guarda en un variable string, seguidamente esa variable es guardada en un archivo con extensión .gtE

```
1 referencia
private void exportarErrorBoton_Click(object sender, EventArgs e)
{
    int counter = 0;
    string datos = "";

    for (int i = 0; i < dataGridView2.Rows.Count - 1; i++)
    {
        datos += "-> ERROR # " + counter + ": \n\t";
        counter++;
        for (int j = 0; j < dataGridView2.Columns.Count; j++)
        {
            datos += dataGridView2.Rows[i].Cells[j].Value.ToString();
            if (j < dataGridView2.Columns.Count - 1)
                datos += ",\t";
        }
        datos += "\n";
    }

    manejador.guardarErrorComo(saveFileDialog1, datos);


}
```

2. Partes esenciales de la interface grafica del IDE

Implementar en C# un IDE cuenta con ciertas partes esenciales como:

- Un form para el manejo de archivos donde se podrán crear proyectos o abrir los proyectos que ya fueron creados con anterioridad.

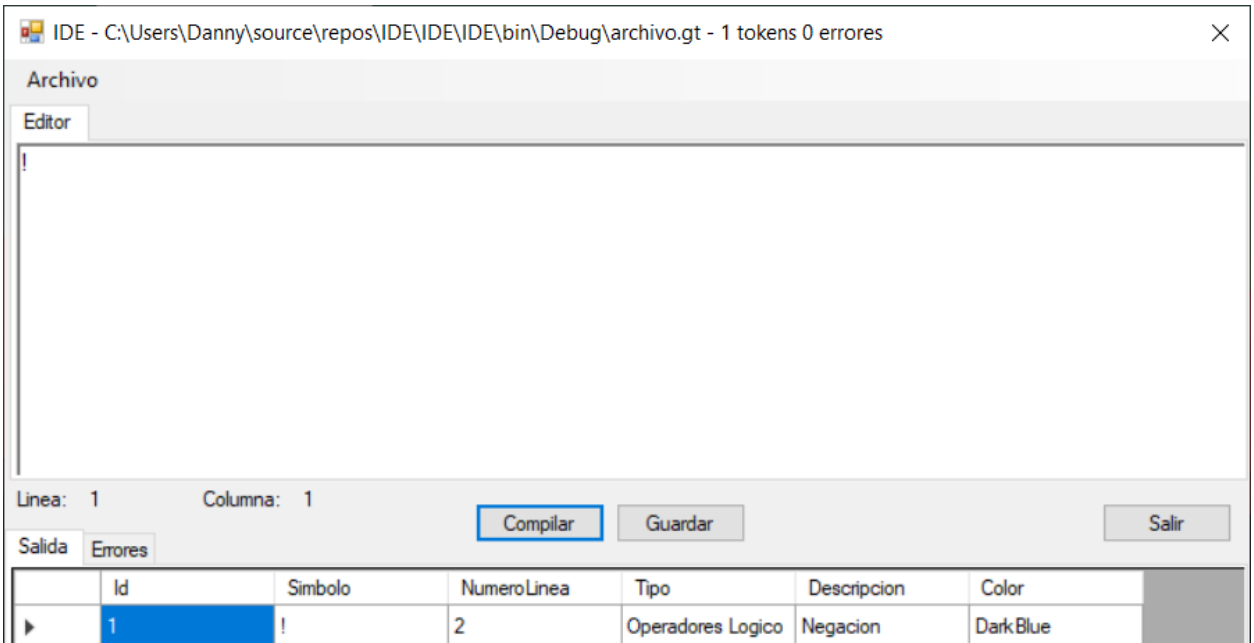
- Un richTexbox donde los desarrolladores podrán escribir su código de manera ordenada para las aplicaciones que estos tengan que realizar.

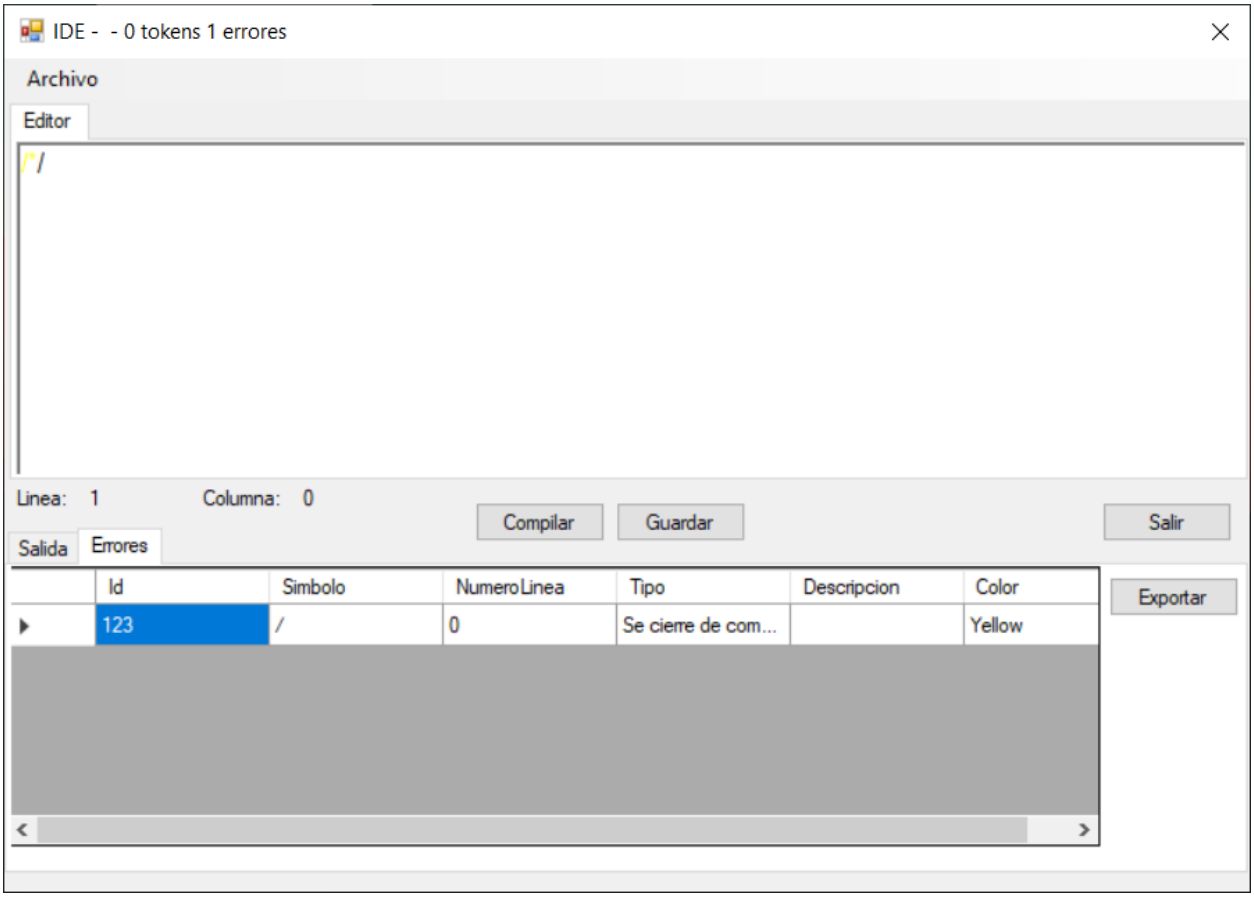
- Una etiqueta donde muestre la fila y la columna donde se encuentre el cursor en cualquier momento

- Botón de compilación analiza y pinta el código escrito por el usuario, y muestran en una tabla los tokens resultados del análisis, y si hay errores los muestra en otra tabla de errores.

- Botón para guardar sobre escribe el archivo abierto con los cambies realizados por el usuario.

Botón tool strip menú que tiene los siguientes menús: Nuevo, Guardar Como, Cargar.




IDE - - 0 tokens 1 errores
Archivo
Editor
/*/
Linea: 1
Columna: 0
Compilar
Guardar
Salir
Salida
Errores
Id
Simbolo
NumeroLinea
Tipo
Descripcion
Color
123
/
0
Se cierre de com...
Yellow
Exportar

Autómatas