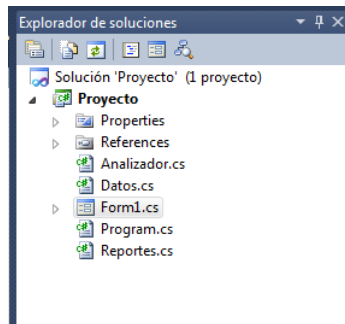


MANUAL TECNICO



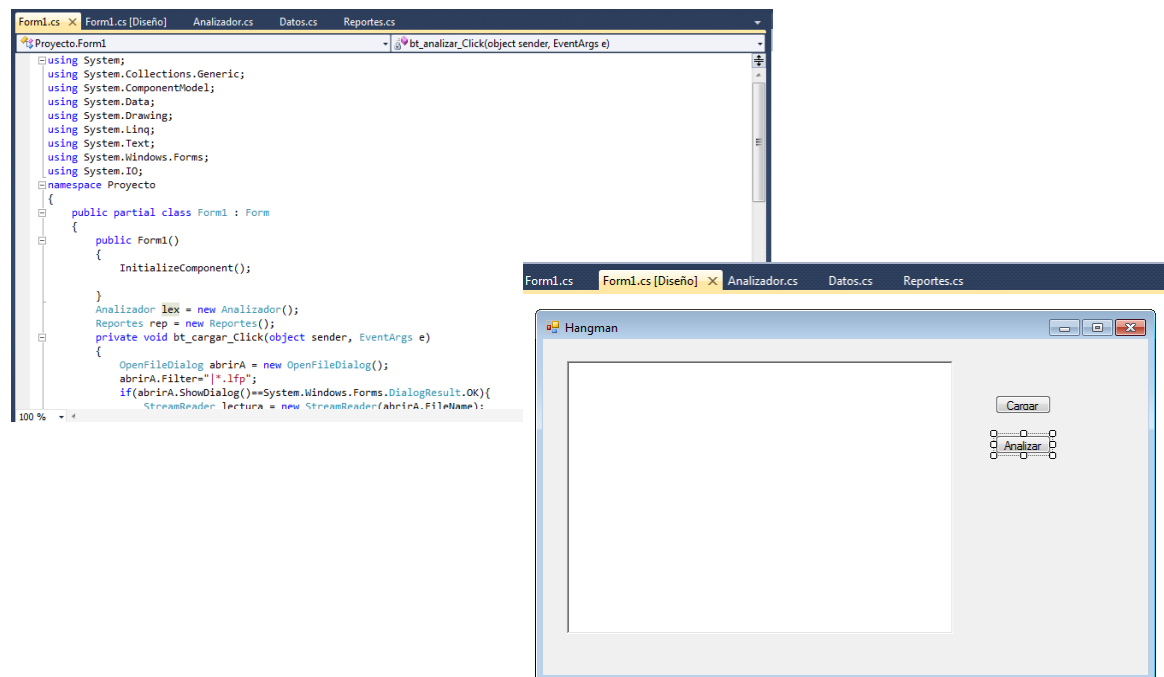
Composición básica de la aplicación

La aplicación de escritorio esta distribuida de la siguiente forma:



Un paquete principal que contiene todas las clases que se requieren para implementar la solución. Esta a su vez contiene cuatro clases que se detallaran a continuación:

- Form1.cs: en esta se encuentra el método main y en esta se cargan la interfaz de la aplicación



- Analizador.cs: En esta se encuentran todas las instrucciones para poder generar el análisis léxico, además esta contiene una serie de métodos adicionales.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Collections;

namespace Proyecto
{
    class Analizador
    {
        int state = 0;
        string lexema = "";
        int num = 0;
        int numerr = 0;
        int fila = 1;
        int columna = 0;

        public List<Datos> arregloToken = new List<Datos>();
        public List<Datos> arregloError = new List<Datos>();

        public List<Datos> getArregloToken()
        {
            return arregloToken;
        }

        //-----Método analizar-----
        public string lexico(string entrada)
        {
            string texto = entrada;
            Char[] cadena = texto.ToCharArray();
            for (int i = 0; i < cadena.Length; i++)
            {
                if (cadena[i] == 10)
                {
                    fila++;
                    columna = 0;
                }
                else
                {
                    columna++;
                }

                switch (state)
                {
                    //-----Estado 0-----
                    case 0:
                        if (Char.IsLetter(cadena[i]))
                        {
                            break;
                        }
                        //-----Estado 1-----
                        case 1:
                            if (Char.IsLetter(cadena[i]))
                            {
                                lexema = lexema + cadena[i];
                                state = 1;
                            }
                            else if (Char.IsDigit(cadena[i]))
                            {
                                lexema = lexema + cadena[i];
                                state = 1;
                            }
                            else if (cadena[i] == 38)
                            {
                                lexema = lexema + cadena[i];
                                state = 4;
                            }
                            else if (cadena[i] == 32 || cadena[i] == 10 || cadena[i] == 9)
                            {
                                num++;
                                arregloToken.Add(new Datos(num, lexema, tipoT(lexema), fila, columna));
                                lexema = lexema + cadena[i];
                                state = 100;
                            }
                            break;
                        //-----Estado 11-----
                        case 11:
                            if (Char.IsLetter(cadena[i]))
                            {
                                lexema = lexema + cadena[i];
                                state = 11;
                            }
                            else if (Char.IsDigit(cadena[i]))
                            {
                                lexema = lexema + cadena[i];
                                state = 11;
                            }
                            else if (cadena[i] == 32 || cadena[i] == 10 || cadena[i] == 9)
                            {
                                num++;
                                arregloToken.Add(new Datos(num, lexema, tipoT(lexema), fila, columna));
                                lexema = "";
                                state = 0;
                            }
                            break;
                    }
                }
            }

            StreamWriter archivo = new StreamWriter("nom4");
            archivo.WriteLine(cod);
            archivo.Close();
        }

        //-----Método Verificar tokens-----
        public string tipoT(string lexema)
        {
            string tiptoken;

            switch (lexema)
            {
                case "Configuracion":
                    tiptoken = "Token_Configuracion";
                    break;
                case "configuracion":
                    tiptoken = "Token_Configuracion";
                    break;
                case "Juego":
                    tiptoken = "Token_Juego";
                    break;
                case "Niveles":
                    tiptoken = "Token_Niveles";
                    break;
            }
        }
    }
}

```

- Datos.cs: Aquí se crean los constructores que almacenan los distintos atributos que contendrá nuestra lista de objetos.
También consta de sus respectivos métodos get y set de cada atributo.

```

Form1.cs  Form1.cs [Diseño]  Analizador.cs  Datos.cs  Reportes.cs
Proyecto.Datos
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Proyecto
{
    class Datos
    {
        int numero;
        int fila = 1;
        int columna;

        string lexi;
        string error;
        string token;
        string Terror = "Error Lexico, Token No reconocido";

        public Datos(int num, string lexe, string tokn, int f, int c){
            this.numero = num;
            this.lexi = lexe;
            this.token = tokn;
            this.fila = f;
            this.columna = c;
        }
    }
}

```

```

        string token;
        string Terror = "Error Lexico, Token No reconocido";

        public Datos(int num, string lexe, string tokn, int f, int c){
            this.numero = num;
            this.lexi = lexe;
            this.token = tokn;
            this.fila = f;
            this.columna = c;
        }

        public Datos(int num, string error, int f, int c)
        {
            this.numero = num;
            this.error = error;
            this.fila = f;
            this.columna = c;
            string tipe = this.Terror;
        }

        //-----Getters y setter-----
        public int getNumero() {
            return numero;
        }

        public void setNumero(int nume) {
            this.numero = nume;
        }
    }
}

```

```

        public void setNumero(int nume) {
            this.numero = nume;
        }

        public int getFila()
        {
            return fila;
        }

        public void setFila(int fila)
        {
            this.fila = fila;
        }

        public int getColumn()
        {
            return columna;
        }

        public void setColumn(int colum)
        {
            this.columna = colum;
        }

        public string getLexi()
        {
            return lexi;
        }
    }
}

```

- Reportes.cs: clase encargada de crear un archivo html, para luego llenarlo con los datos de la lista de objetos estos archivos corresponden a la tabla de símbolos y errores.

```

Form1.cs  Form1.cs [Diseño]  Analizador.cs  Datos.cs  Reportes.cs X
Proyecto.Reportes
using System.Text;
using System.Collections;
using System.IO;
using System.Windows.Forms;

namespace Proyecto
{
    class Reportes
    {
        Analizador an=new Analizador();
        List<Datos> listok;
        List<Datos> lise;

        public void CrearRT(List<Datos> lista) {
            listok=lista;
            String html = "<html>\n"
                + "<head>\n"
                + "<style type='text/css'>\n" // style de css
                + "table {\n" +
                "    font-family: verdana,arial,sans-serif;\n" +
                "    font-size:11px;\n" +
                "    color:#333333;\n" +
                "    border-width: 1px;\n" +
                "    border-color: #666666;\n" +
                "    border-collapse: collapse;\n" +
                "    width: 100%;\n" +

```

```

+ "<th>Componente léxico</th>\n"
+ "<th>File</th>\n"
+ "<th>columna</th>\n"
+ "</tr>\n";

        for (int i = 0; i < listok.Count; i++)
        {
            html += "<tr>\n"
                + "<td>" + ((Datos)listok[i]).getNumero() + "</td>\n"
                + "<td>" + ((Datos)listok[i]).getLexi() + "</td>\n"
                + "<td>" + ((Datos)listok[i]).getToken() + "</td>\n"
                + "<td>" + ((Datos)listok[i]).getFile() + "</td>\n"
                + "<td>" + ((Datos)listok[i]).getColumn() + "</td>\n"
                + "</tr>\n";
        }

        html+="

```

```

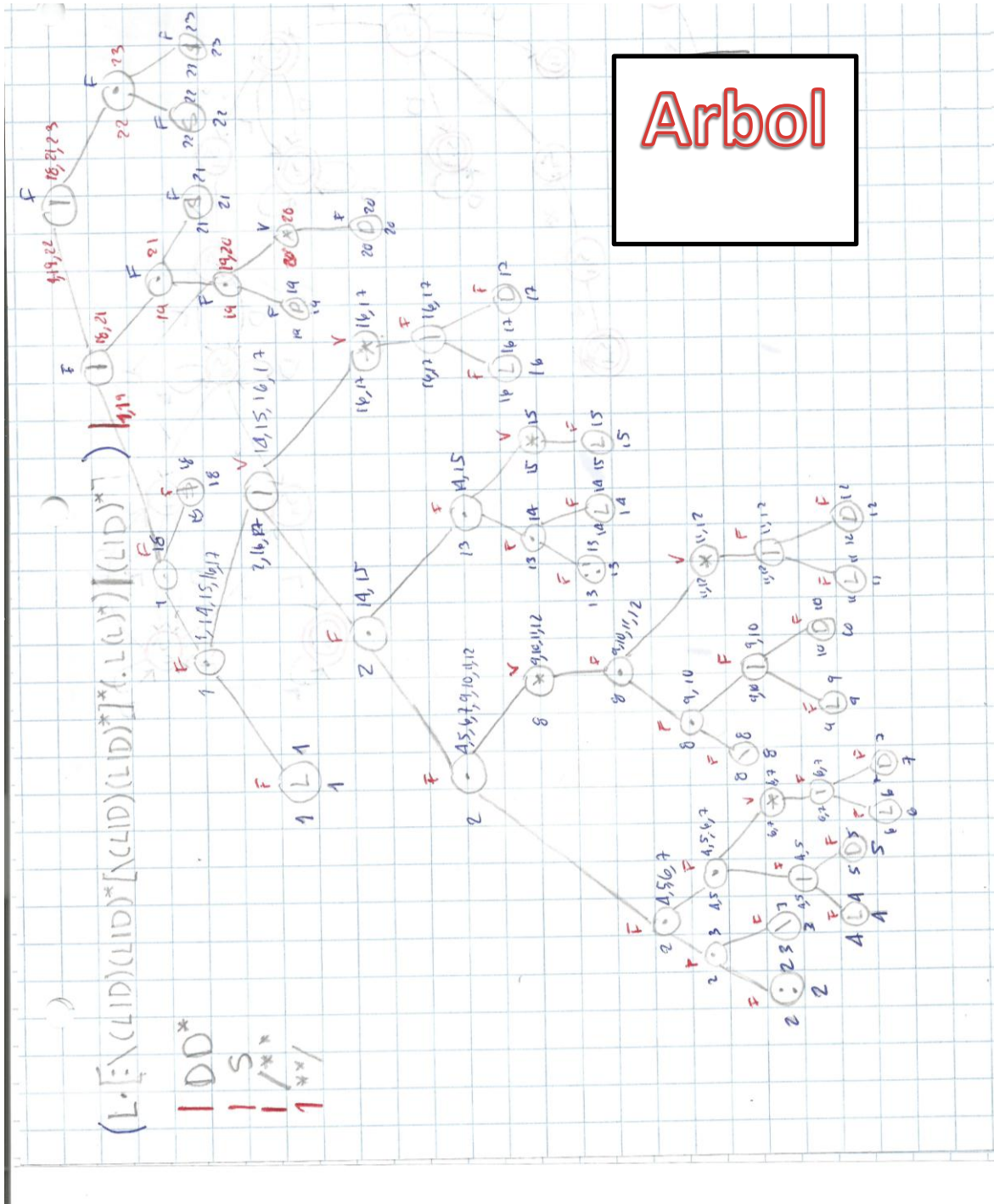
+ "</html>";

        MessageBox.Show("Archivo Creado");
        an.Archivo(html, "Símbolos");
    }

    public void CrearRE(List<Datos> lista)
    {
        lise = lista;
        String html2 = "<html>\n"
            + "<head>\n"
            + "<style type='text/css'>\n" // style de css
            + "table {\n" +
            "    font-family: verdana,arial,sans-serif;\n" +
            "    font-size:11px;\n" +
            "    color:#333333;\n" +
            "    border-width: 1px;\n" +
            "    border-color: #666666;\n" +
            "    border-collapse: collapse;\n" +
            "    width: 100%;\n" +
            "}\n"
            + "th {\n" +
            "    border-width: 1px;\n" +
            "    padding: 8px;\n" +
            "    border-style: solid;\n" +

```

A continuación se muestra la implementación del método del árbol para poder generar el autómata que se utilizó para la programación del analizador.



Arbol

Tabla de siguientes

	$\text{Sig}(i)$	
1	2, 16, 17, 18	$S_0 = \{1, 19, 22\}$
2	3	$\text{Sig}(L) = \text{Sig}(1) = \{2, 16, 17, 18\}$
3	4, 5	$\text{Sig}(D) = \text{Sig}(19) = \{20, 21\}$
4	6, 7, 8, 13	$\text{Sig}(S) = \text{Sig}(1) = \{8, 3\}$
5	6, 7, 8, 13	$S_1 = \{2, 16, 17, 18\}$
6	6, 7, 8, 13	$\text{Sig}(L) = \text{Sig}(2) = \{3\}$
7	6, 7, 8, 13	$S_2 = \{20, 21\}$
8	9, 10	$\text{Sig}(D) = \text{Sig}(20) = \{20, 21\}$
9	8, 11, 12, 13	$S_3 = \{2, 3\}$
10	8, 11, 12, 13	$S_4 = \{3\}$
11	8, 11, 12, 13	$\text{Sig}(L) = \text{Sig}(4, 5) = \{4, 5\}$
12	8, 11, 12, 13	$S_5 = \{4, 5\}$
13	14	$\text{Sig}(L) = \text{Sig}(14) = \{6, 7, 8, 13\}$
14	15, 18	$\text{Sig}(D) = \text{Sig}(15) = \{6, 7, 8, 13\}$
15	15, 18	$S_6 = \{6, 7, 8, 13\}$
16	16, 17, 18	$\text{Sig}(L) = \text{Sig}(16) = \{6, 7, 8, 13\}$
17	16, 17, 18	$\text{Sig}(D) = \text{Sig}(17) = \{6, 7, 8, 13\}$
18		$\text{Sig}(L) = \text{Sig}(18) = \{9, 10\}$
19	20, 21	$\text{Sig}(D) = \text{Sig}(19) = \{14, 18\}$
20	20, 21	$S_7 = \{9, 10\}$
21		$\text{Sig}(L) = \text{Sig}(20) = \{8, 11, 12, 13\}$
22	23	$\text{Sig}(D) = \text{Sig}(21) = \{8, 11, 12, 13\}$
23		$S_8 = \{14\}$
		$\text{Sig}(L) = \{15, 18\}$
		$S_9 = \{8, 11, 12, 13\}$
		$\text{Sig}(L) = \{9, 10\}$
		$\text{Sig}(L) = \{11\} = \{8, 11, 12, 13\}$
		$\text{Sig}(D) = \{17\} = \{8, 11, 12, 13\}$
		$\text{Sig}(L) = \{13\} = \{14\}$
		$S_{10} = \{15, 18\}$
		$\text{Sig}(L) = \{15\} = \{15, 18\}$

Autómata Determinístico

