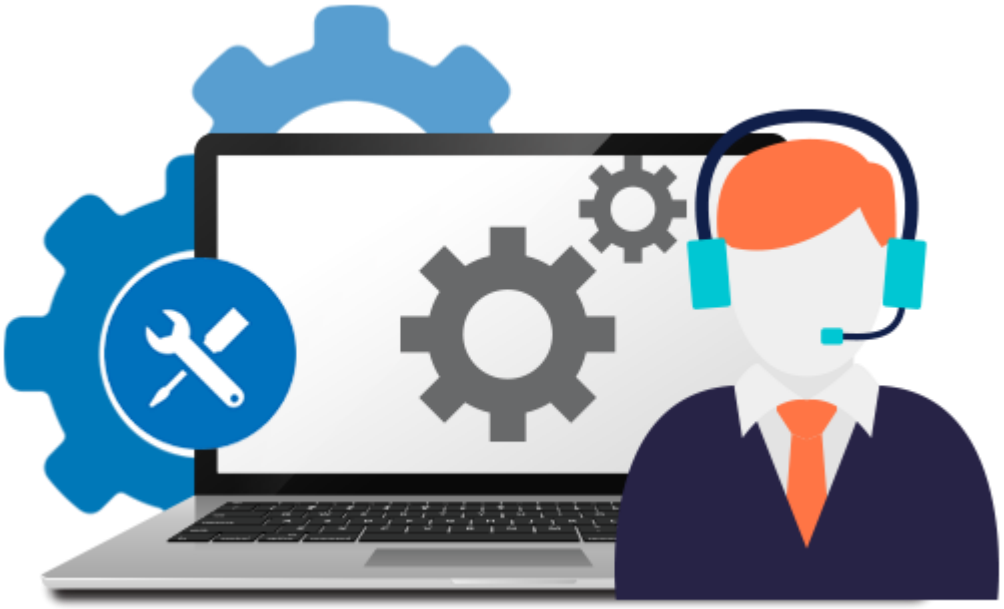
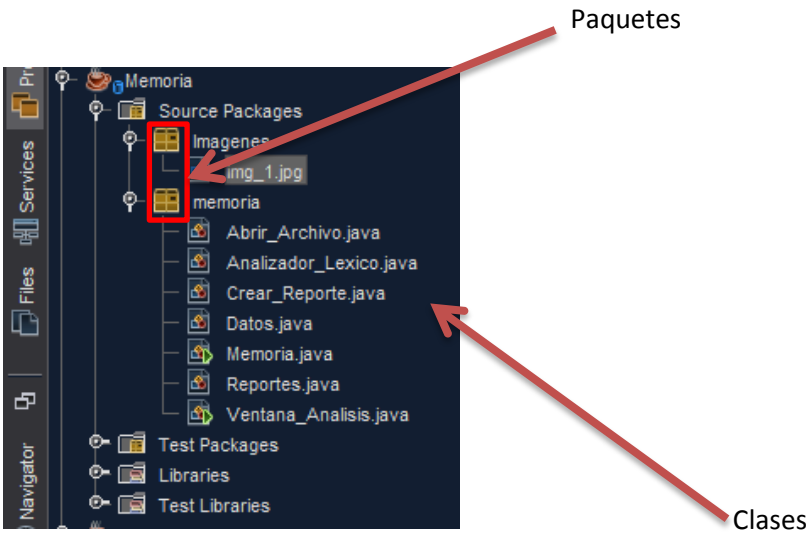
MANUAL TECNICO



Composición básica de la aplicación:

La aplicación de escritorio esta distribuida de la siguiente forma:



Paquete memoria: Esta contiene el método main y las distintas clases que conforman la funcionalidad de la aplicación.

A continuación se detallara la función de cada una de las clases:

- Abrir_Archivo.java

```
6
7   public class Abrir_Archivo {
8       FileInputStream entrada;
9       FileOutputStream salida;
10      File Archivo;
11
12      public String Abrir(File Archivo){
13          String cadena="";
14          try{
15              entrada=new FileInputStream(Archivo);
16              int asc=0;
17              while((asc=entrada.read())!=-1){
18              char caract=(char)asc;
19              cadena+=caract;
20              }
            }catch(Exception e){
```

- **Analizador_lexico.java:** contiene el método para analizar un archivo. Recibe como parámetro un valor tipo String.

```
    }

    //----------------Inicio-------------------------
    public void lexico(String entrada) {
       entrada += " ";
        char[] caracter = entrada.toCharArray();
        //----------loop que recorre el arreglo-------------
        for (int i = 0; i < caracter.length; i++) {
            //-----------Fila y columna---------------------
            switch (caracter[i]) {
                case 10:
                    fila++;
                    columna = 0;
                    break;
                case 9:
                    columna+=5;
```

Para realizar el análisis se utiliza un switch, este nos servirá para recorrer el autómata que se está implementando. (Revisar Anexos)

```
//-------------Inicio_Analisis-----------------------------
switch (estado) {
    //---------Estado 0-------------------------------------
    case 0:
        //-------letra-----------
        if (((caracter[i] >= 65 && caracter[i] <= 90)) || ((caracter[i] >= 97 && caracter[
            estado = 1;
            lexema += caracter[i];
        } //-------Numero-----------
        else if ((caracter[i] >= 48 && caracter[i] <= 57)) {
            estado = 2;
            lexema += caracter[i];
        } else if (caracter[i] == 45) {
            estado = 3;
            lexema += caracter[i];
```

Se compara el carácter con un valor del código ASCII

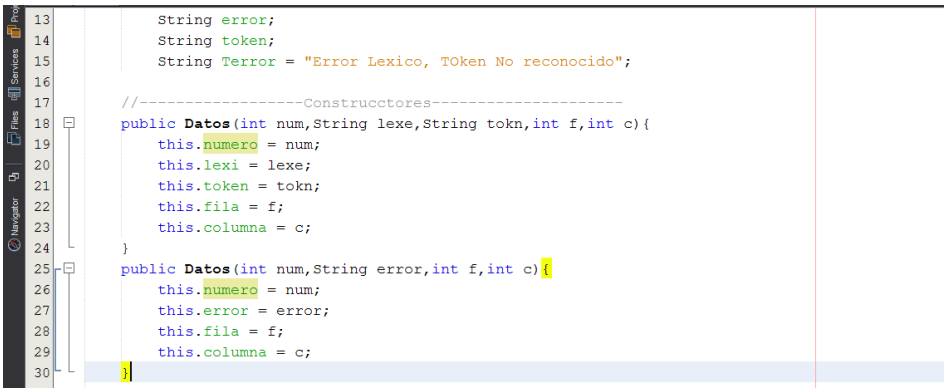
Dentro de esta clase también se encuentra el método que nos servirá para reconocer que tipo de token es el lexema reconocido.

```
//--------Metodo Para Reconocer Tokens---------------------------
public String tipoT(String lexema) {
    String tipo_token = " ";

    if ((lexema.equals("Configuracion")) || (lexema.equals("configuracion"))) {
        tipo_token = "Token_Configuracion";
    } else if ((lexema.equals("Juego")) || (lexema.equals("juego"))) {
        tipo_token = "Token_Juego";
    } else if ((lexema.equals("Nivel")) || (lexema.equals("nivel"))) {
        tipo_token = "Token_Nivel";
    } else if ((lexema.equals("Facil")) || (lexema.equals("facil"))) {
        tipo_token = "Token_Facil";
    } else if ((lexema.equals("Intermedio")) || (lexema.equals("intermedio"))) {
        tipo_token = "Token_Intermedio";
    } else if ((lexema.equals("Dificil")) || (lexema.equals("dificil"))) {
```

- Crear_Reporte.java: Este contiene un método que se encargara de generar el archivo HTML en la ruta especificada.
  Están compuestas por dos parámetros de tipo String un parámetro guarda la plantilla html y el otro el tipo de archivo a generar.

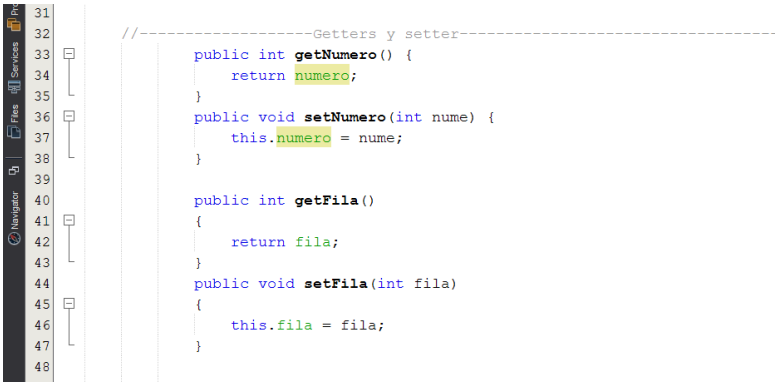
```
public void guardarReporte(String codigoHTML, String tipoReporte){

    String nombreArchivo = "";
    if(tipoReporte == "Simbolos"){
        nombreArchivo="C:\\Users\\Dell E5420\\Documents\\Simbolos.html";

    }else if(tipoReporte == "Error"){
        nombreArchivo="C:\\Users\\Dell E5420\\Documents\\Errores.html";
    }
    archivo = new File(nombreArchivo);
    try(FileWriter escritura = new FileWriter(archivo)){
        escritura.write(codigoHTML);
        System.out.println("sdkflkasmdlkf");
    }catch(IOException e){
        e.printStackTrace();
    }
```

- **Datos.java**: Este contiene dos constructores, cada uno de ellos con distintos parámetros. Se usara la información de los constructores para generar dos listas que almacenaran la tabla de símbolos y la de errores.



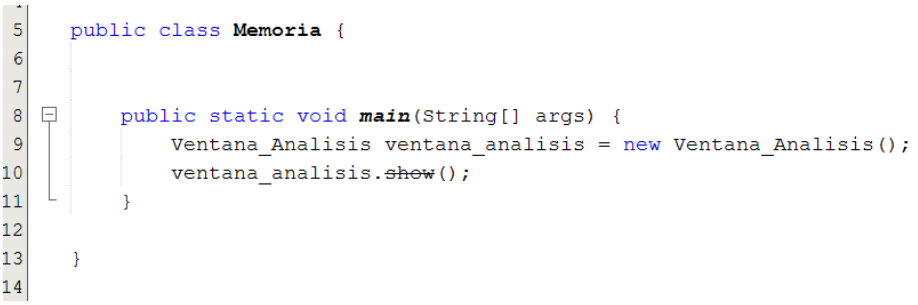
```
13          String error;
14          String token;
15          String Terror = "Error Lexico, TOken No reconocido";
16
17      //-----------------Construcctores--------------------
18      public Datos(int num,String lexe,String tokn,int f,int c){
19          this.numero = num;
20          this.lexi = lexe;
21          this.token = tokn;
22          this.fila = f;
23          this.columna = c;
24      }
25      public Datos(int num,String error,int f,int c){
26          this.numero = num;
27          this.error = error;
28          this.fila = f;
29          this.columna = c;
30      }
```

Además contiene los getter y setters encargados de obtener los datos que se guardan en las listas.



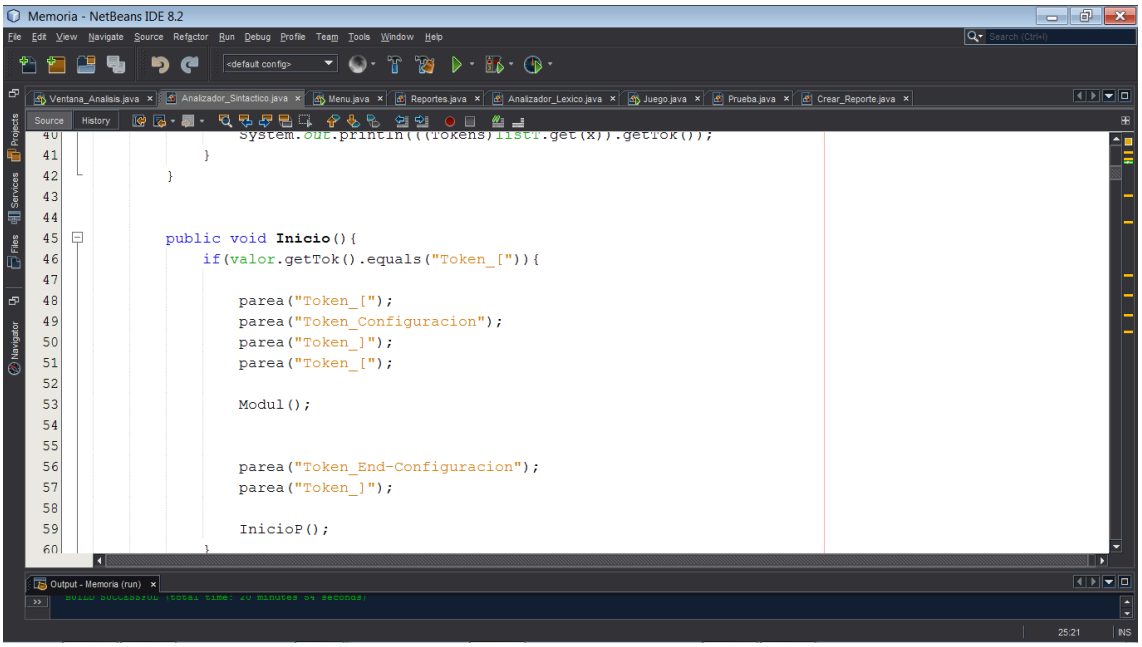
```
31
32      //-----------------Getters y setter-----------------------------------
33          public int getNumero() {
34              return numero;
35          }
36          public void setNumero(int nume) {
37              this.numero = nume;
38          }
39
40          public int getFila()
41          {
42              return fila;
43          }
44          public void setFila(int fila)
45          {
46              this.fila = fila;
47          }
48
```

- **Memoria.java:** esta contiene el main del programa y se encarga de ejecutar un form.

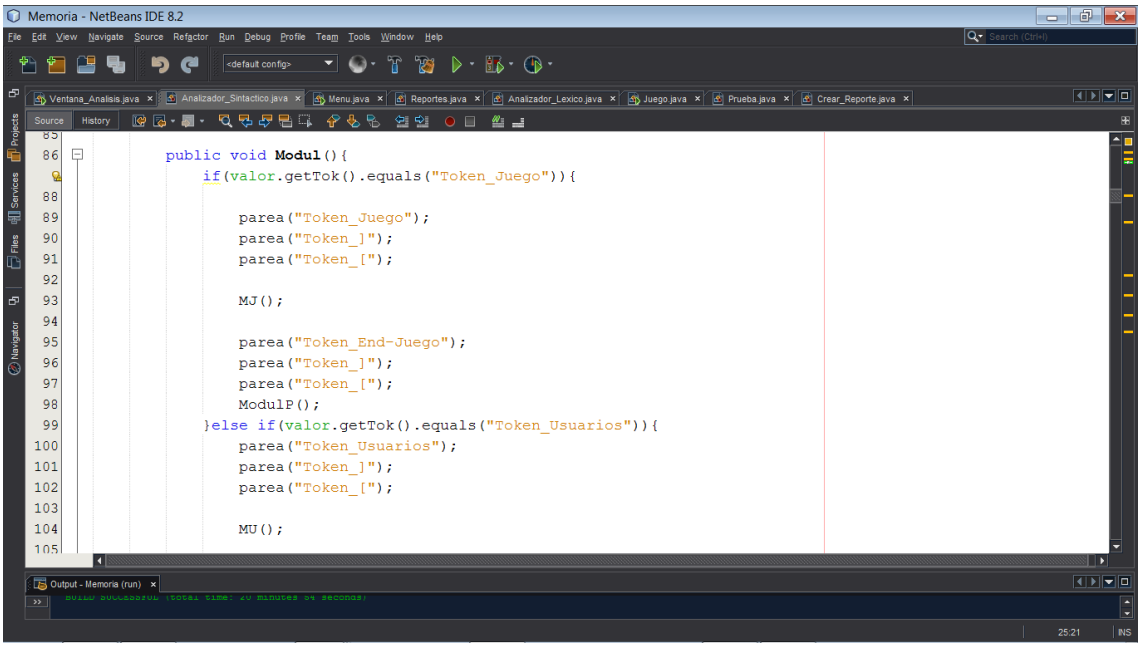


```
5   public class Memoria {
6
7
8       public static void main(String[] args) {
9           Ventana_Analisis ventana_analisis = new Ventana_Analisis();
10          ventana_analisis.show();
11      }
12
13  }
14
```

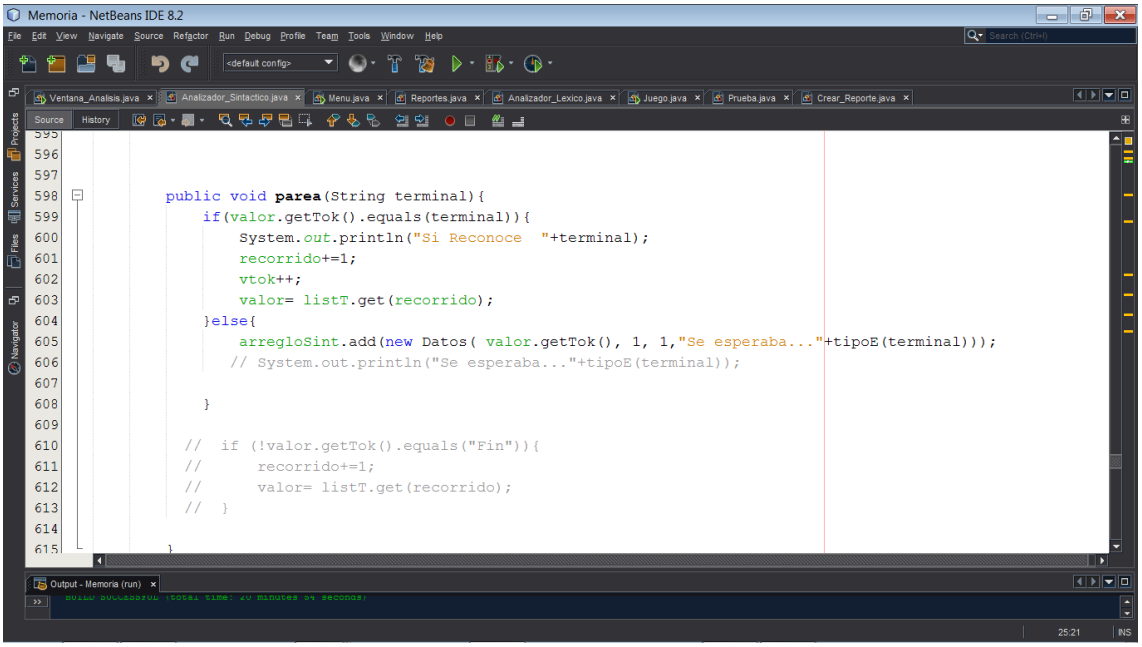
- Clase Analizador Sintactico: esta contiene la implementación de la gramatica encargada de reconocer el lenguaje(ver anexo)



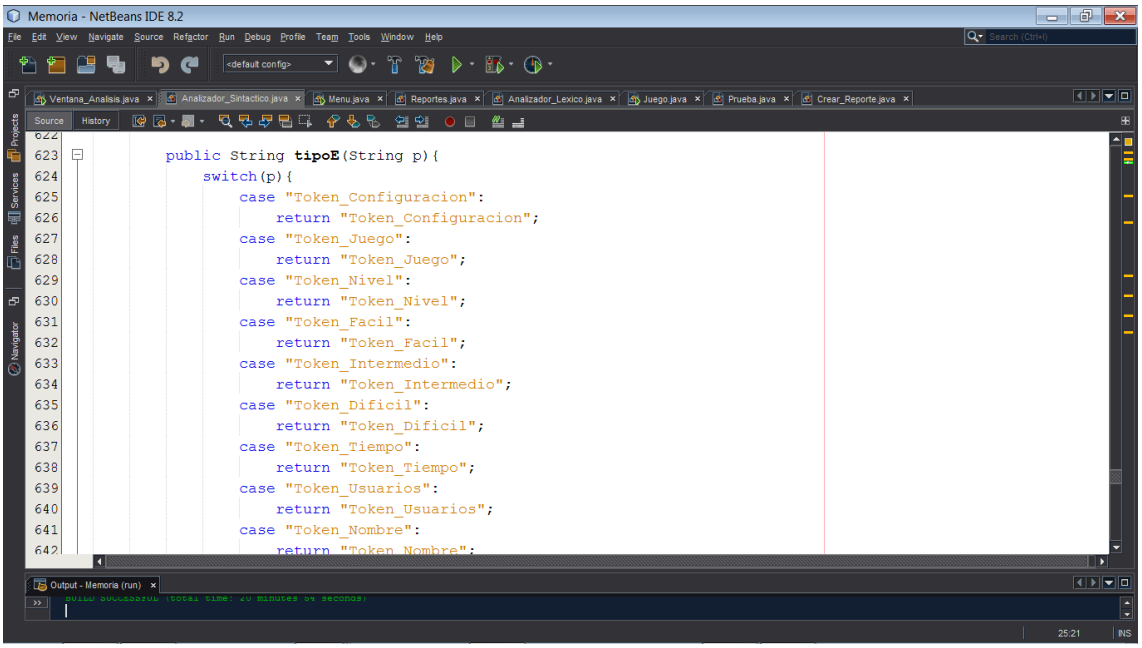
Cada producción de la gramatica se considera un método el cual contendrá el llamado a terminales o inclusive no terminales.



El método pare a es el encargado de comprobar si el valor que se envie como parámetro concuerda con la gramatica



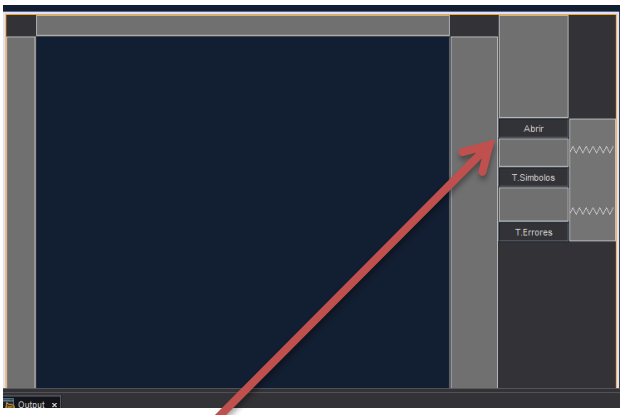
Tambien estableci un método que me va a devolver el tipo de token encontrado para mostrarlo en la tabla de errores



- Reportes.java: Clase encargada de generar el código html que contendrá el valor de los arrays y le mandara como parámetros un string a la clase Crear_Reportes.

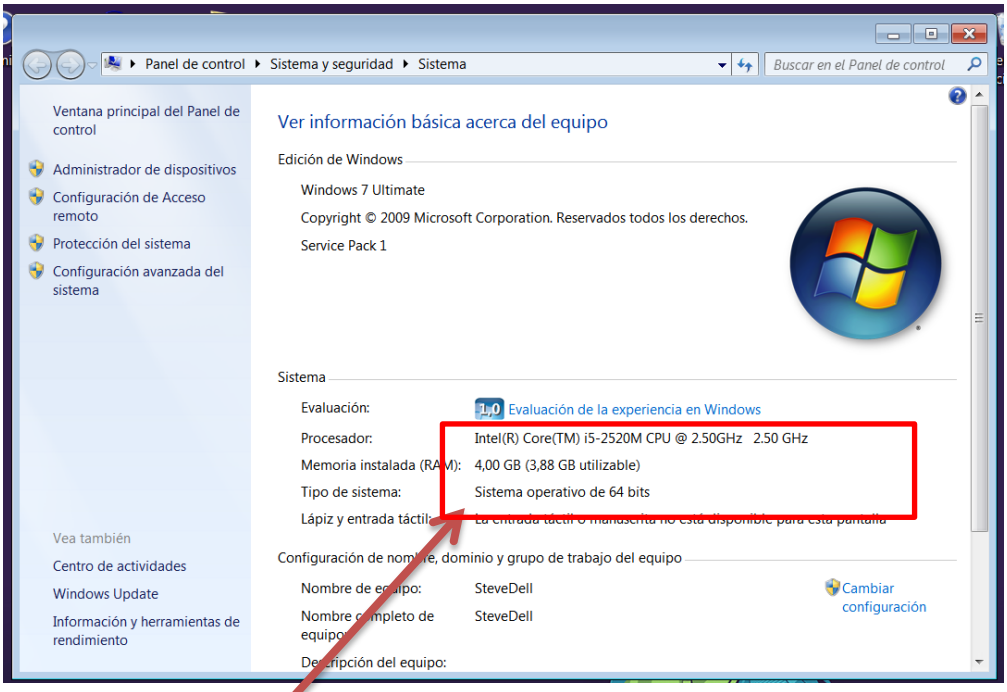
```
10      //------Declaracion de listas-----------------------
11      ArrayList<Datos>list;
12      ArrayList<Datos>lise;
13      Crear_Reporte crearR=new Crear_Reporte();
14      //---------Metodo para Reporte Tokens---------------
15      public void CrearRT(ArrayList lista){
16          list=lista;
17
18          //------------------HTML-----------------------------------------------
19              html = "<html>\n"
20                  +"<head>\n"
21                 +"<style type=\"text/css\">\n" //_____________________________style de css_____________________
22                 +"table {\n" +
23                  "       font-family: verdana,arial,sans-serif;\n" +
24                  "       font-size:11px;\n" +
25                  "       color:#333333;\n" +
26                  "       border-width: 1px;\n" +
27                  "       border-color: #666666;\n" +
```

- **Ventana_Analisis.java**: Contiene el form de la aplicación. Tiene la distribución de el cuadro de texto y sus botones




```
 */
@SuppressWarnings("unchecked")
Generated Code

private void btn_analizarActionPerformed(java.awt.event.ActionEvent evt) {
    if(opcion){

        if (selec.showDialog(this, "Abrir archivo")== JFileChooser.APPROVE_OPTION){
            archiv=selec.getSelectedFile();
            if(archiv.getName().endsWith("lfp")){
                String contenido=marchi.Abrir(archiv);
                Area_editor.setText(contenido);
            }else{
                JOptionPane.showMessageDialog(rootPane, "NO es un archivo .lfp");
            }

        }
```

Hardware donde se realizó el proyecto.



Requerimientos minimos

Software Utilizado

IDE Utilizado

Anexo

Árbol

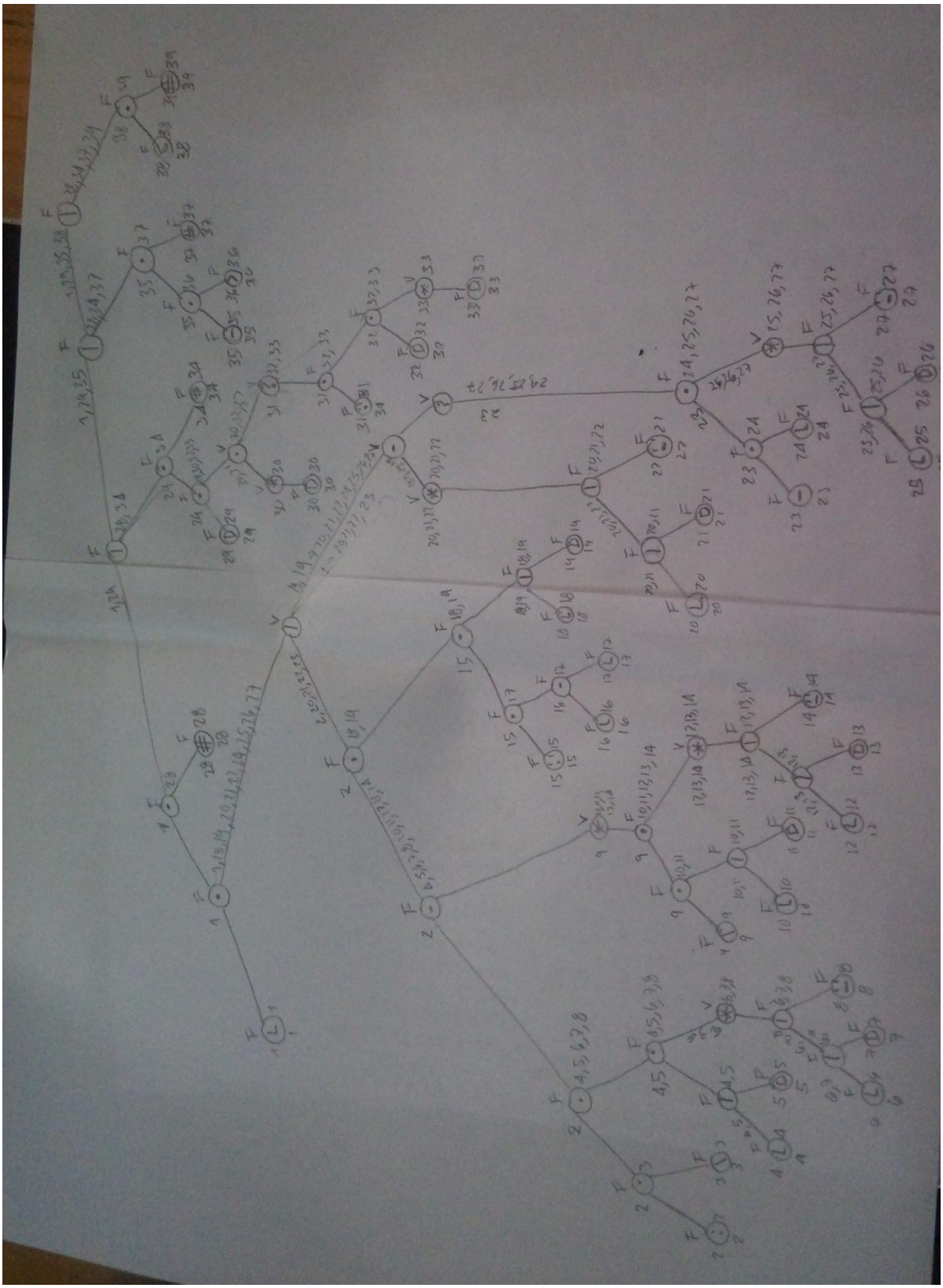
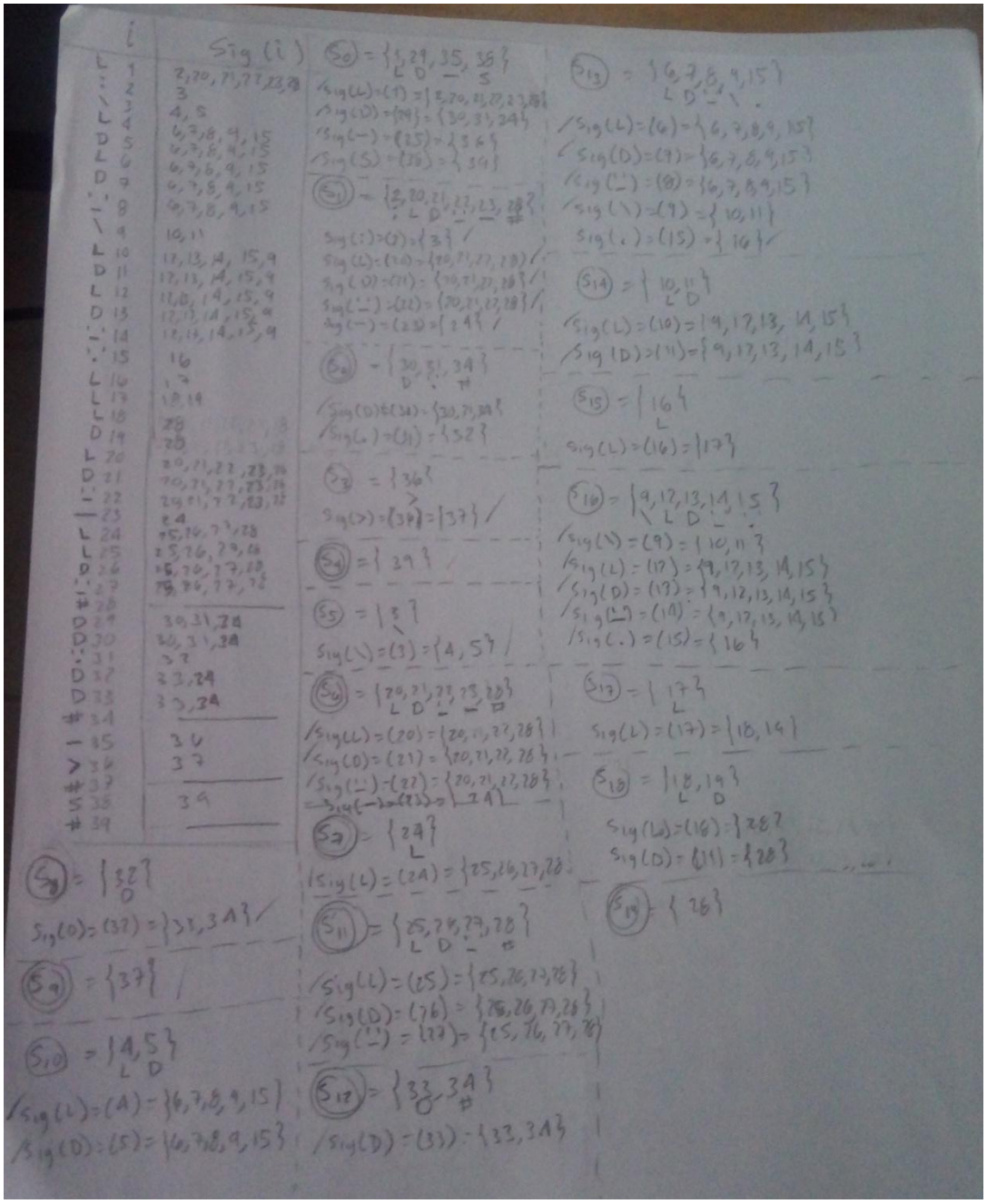
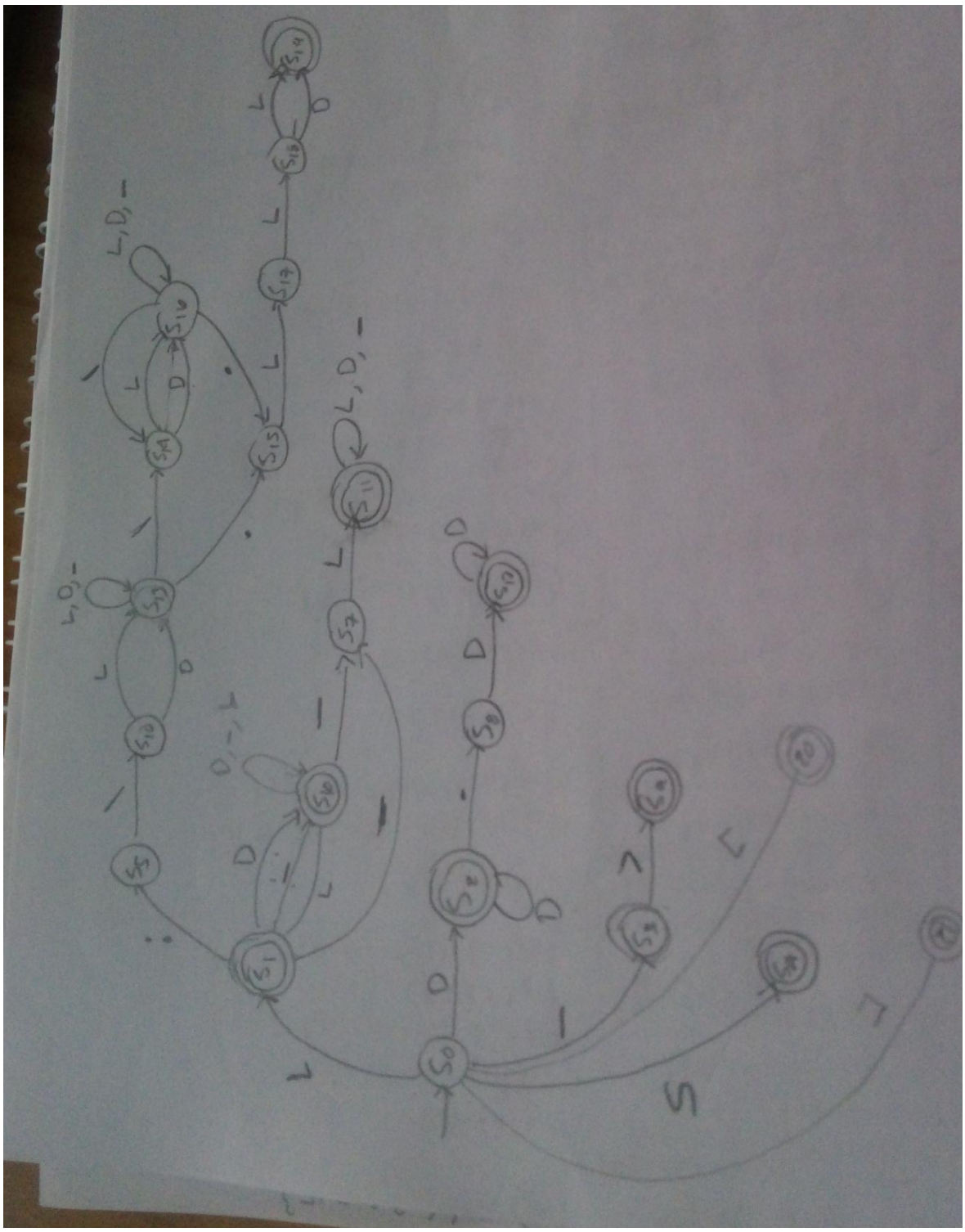


Tabla de siguientes

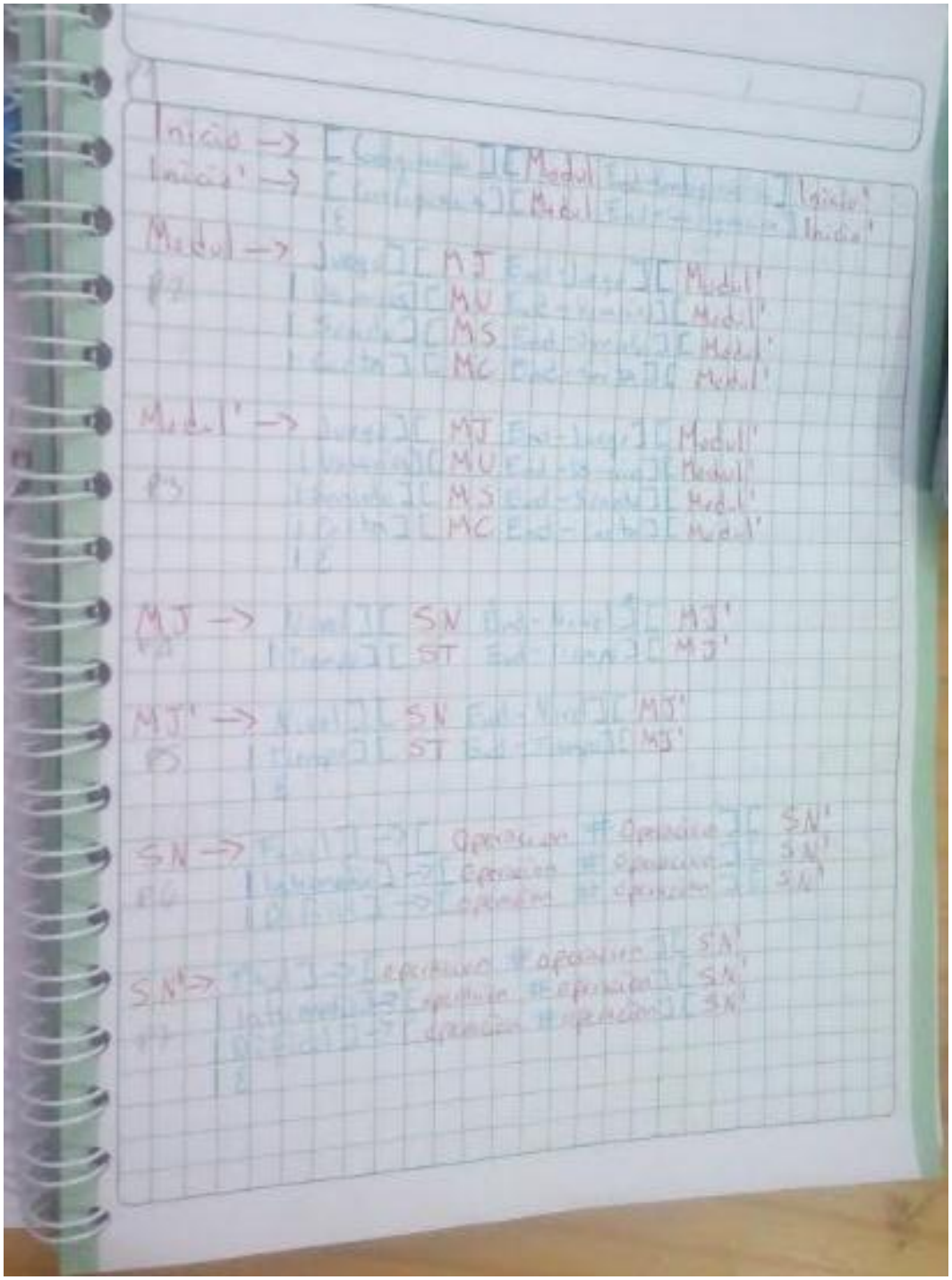


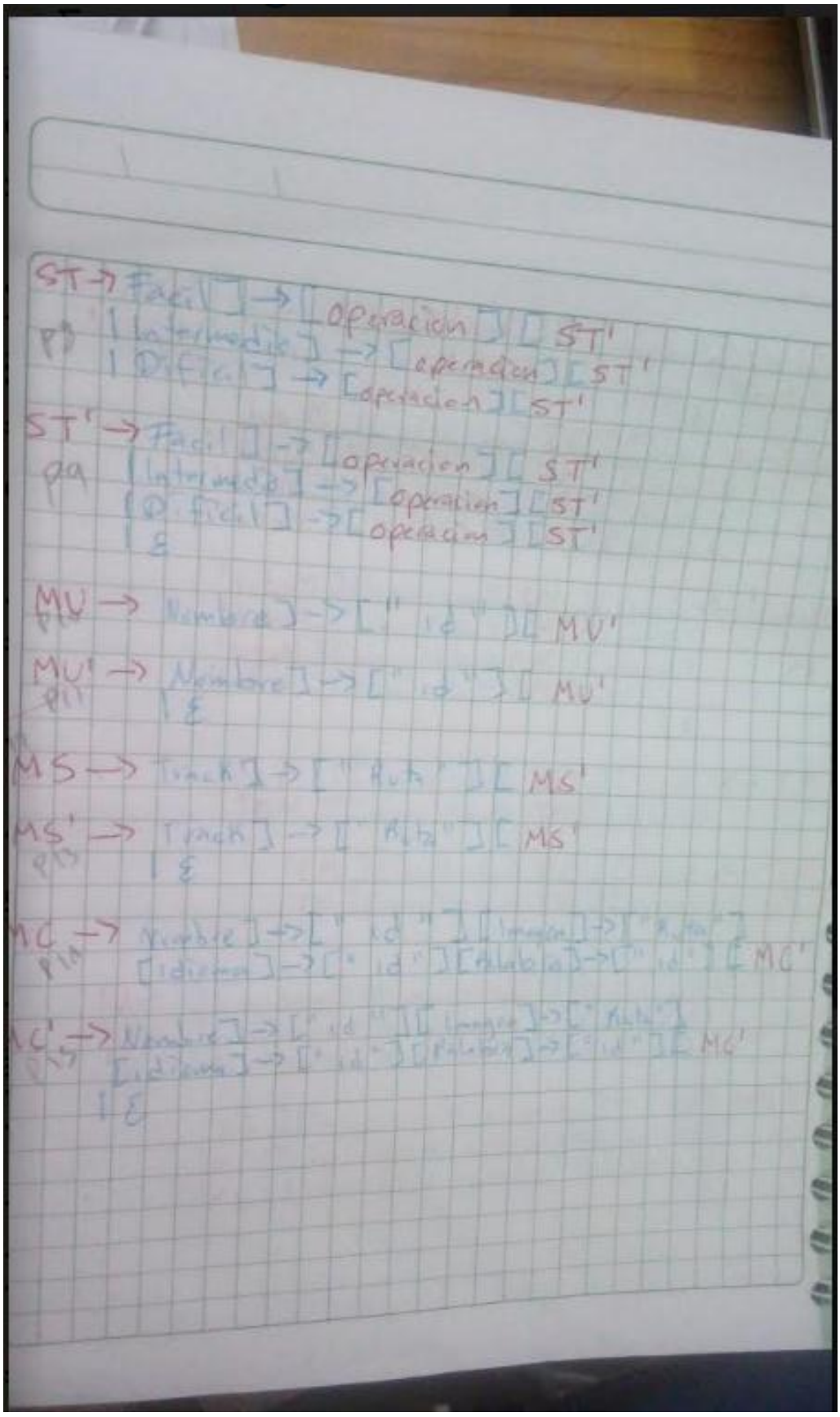
Autómata



Metodo LL!

Gramatica





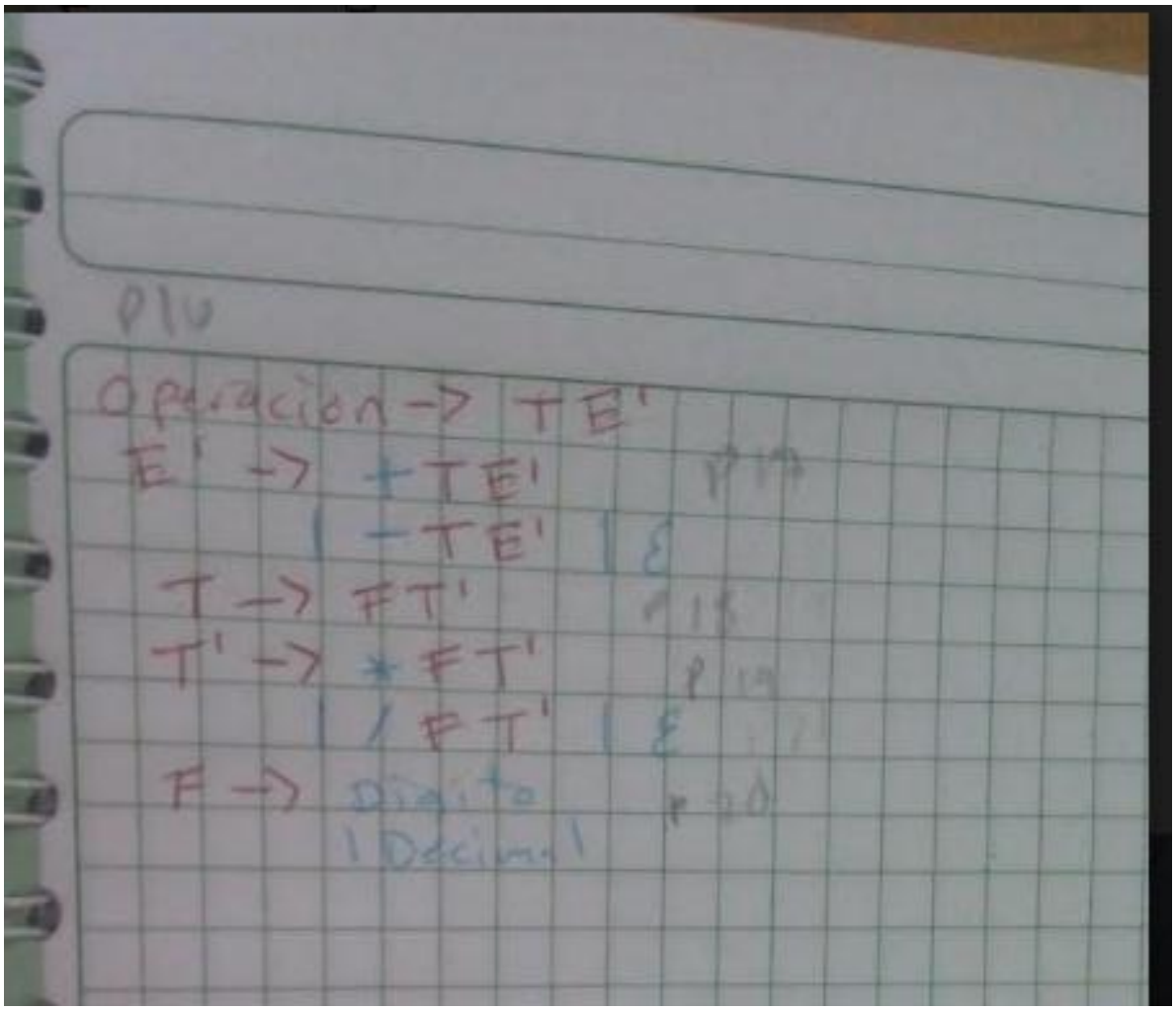

PLU
Operacion -> TE'
E' -> +TE'
| -TE' | ε
T -> FT'
T' -> *FT'
| /FT' | ε
F -> Digito
| Decimal

Tabla primeros y siguientes