MANUAL TECNICO

A continuación se le presenta el manual técnico del programa, con los requerimientos mínimos y el entorno de desarrollo así como las herramientas utilizadas para la creación de dicho programa. Puntos importantes a considerar en el código, para su buen funcionamiento y darle un soporte adecuado. En resumen, el programa es un traductor de texto, entrada en formato uweb y salida en formato html, de un lenguaje llamado Hscript con Struct.

PLATAFORMA DE DESARROLLO

El programa fue desarrollado bajo el sistema operativo Windows 10 de 64 bits, en el IDE Netbeans en su versión 8.0.2, con ayuda del editor de texto Atom, bajo el lenguaje de programación Java en su versión mas actualizada a enero de 2019, Java 8 201.

Para ejecutar el análisis del código, tanto léxico como sintáctico se utilizaron dos herramientas aceptadas por Java, Cup y JFlex. A continuación, se detalla un breve resumen de ambas.

JFLEX

Es una herramienta JAVA que permite actuar sobre aquellas cadenas de un fichero de texto que encajan en una expresión regular.

Flex es un meta compilador que permite generar rápidamente analizadores léxicos que se integran con Java.

Jlex es una herramienta desarrollada en Java que toma como entrada un archivo "entrada", con este crea un archivo fuente java entrada.lex.java correspondiente al analizador léxico. En otras palabras, Jlex es un generador de analizadores léxicos. Los analizadores léxicos toman como entrada una cadena de caracteres y la convierten en una secuencia de tokens.

CUP

JAVA CUP es un parser-generador. Es un analizador sintáctico que construye un parser para gramáticas tipo LALR(1), con código de producción y asociación de fragmentos de código JAVA. Cuando una producción en particular es reconocida, se genera un archivo fuente Java, parser.java que contiene una clase parser, con un método Symbol parser().

Para ejecutar ambos programas, basta con crear dos archivos, un Lexer y otro parser, agregar las librerias correspondientes y escribir en un archivo con extensión "bat" lo siguiente:

java -jar JLex/jflex-full-1.7.0.jar Lexer --encoding utf-8

java -jar Cup/java-cup-11b.jar -parser parser Parser

y automáticamente crea dos nuevas clases de java, su lexer y su parser respectivamente, que serán los encargados de realizar el análisis completo del código.

GRAMATICA UTILIZADA:

MAYUSCULAR = TERMINALES

MINUSCULULAS = NO TERMINALES

S::= TEXTO R_COMPI CIERRA_ETIQUETA inner TEXTO SLASH R_COMPI CIERRA_ETIQUETA

inner::= inner t_inner

| t_inner

t_inner::= TEXTO R_CABECERA CIERRA_ETIQUETA inner_cabecera TEXTO SLASH R_CABECERA CIERRA_ETIQUETA {:

| TEXTO R_CUERPO op_cuerpo CIERRA_ETIQUETA inner_cuerpo TEXTO SLASH R_CUERPO CIERRA_ETIQUETA

op_cuerpo::= R_FONDO IGUAL CADENA

| {:/*epsilon*/:}

inner_cabecera::= TEXTO R_TITULO CIERRA_ETIQUETA TEXTO SLASH R_TITULO CIERRA_ETIQUETA

inner_cuerpo::= inner_cuerpo:a tipo_cuerpo

| tipo_cuerpo

tipo_cuerpo::=TEXTO:a R_PARRAFO op_parrafo:b CIERRA_ETIQUETA TEXTO:c SLASH R_PARRAFO CIERRA_ETIQUETA

|TEXTO:a SLASH R_SALTO CIERRA_ETIQUETA

|TEXTO:a R_TABLA op_tabla:b CIERRA_ETIQUETA lista_tabla:c TEXTO:d SLASH R_TABLA CIERRA_ETIQUETA

|TEXTO:a R_IMAGEN list_img:b CIERRA_ETIQUETA TEXTO:c SLASH R_IMAGEN CIERRA_ETIQUETA

|TEXTO:a R_TEXTOA CIERRA_ETIQUETA TEXTO:b SLASH R_TEXTOA CIERRA_ETIQUETA

|TEXTO:a R_TEXTOB CIERRA_ETIQUETA TEXTO:b SLASH R_TEXTOB CIERRA_ETIQUETA

|TEXTO R_BOTON op_boton CIERRA_ETIQUETA TEXTO SLASH R_BOTON CIERRA_ETIQUETA

|TEXTO R_ESPACIO CIERRA_ETIQUETA inner_cuerpo TEXTO SLASH R_ESPACIO CIERRA_ETIQUETA

|ABRE_HS list_hs R_CIERRAHS

op_parrafo::= ID:a IGUAL CADENA:b

|{: /*epsilon*/ :}

op_boton::= ID:a IGUAL CADENA:b ID:c IGUAL CADENA:d

list_img::= list_img:a op_imagen

| op_imagen

op_imagen::= ID:a IGUAL CADENA

| ID:a IGUAL NUM

op_tabla::= ID:a IGUAL ID

| ID:a IGUAL CADENA

| {:/*epsilon*/:}

lista_tabla::= lista_tabla:a TEXTO:b R_FILA CIERRA_ETIQUETA lista_fila:c TEXTO:d SLASH R_FILA CIERRA_ETIQUETA

| TEXTO:a R_FILA CIERRA_ETIQUETA lista_fila:b TEXTO:c SLASH R_FILA CIERRA_ETIQUETA

lista_fila::= lista_fila:a tipo_columna

| tipo_columna

tipo_columna::= TEXTO:a R_COLUMNAC CIERRA_ETIQUETA lista_columna:b TEXTO:c SLASH R_COLUMNAC CIERRA_ETIQUETA

|TEXTO:a R_COLUMNA CIERRA_ETIQUETA lista_columna:b TEXTO:c SLASH R_COLUMNA CIERRA_ETIQUETA

|TEXTO:a R_COLUMNA CIERRA_ETIQUETA TEXTO:c SLASH R_COLUMNA CIERRA_ETIQUETA

|TEXTO:a R_COLUMNAC CIERRA_ETIQUETA TEXTO:c SLASH R_COLUMNAC CIERRA_ETIQUETA

lista_columna::= lista_columna:a op_columna

|op_columna

op_columna::= TEXTO:a R_PARRAFO op_parrafo:b CIERRA_ETIQUETA TEXTO:c SLASH R_PARRAFO CIERRA_ETIQUETA

|TEXTO:a SLASH R_SALTO CIERRA_ETIQUETA

|TEXTO:a R_IMAGEN list_img:b CIERRA_ETIQUETA TEXTO:c SLASH R_IMAGEN CIERRA_ETIQUETA

|TEXTO:a R_BOTON op_boton:b CIERRA_ETIQUETA TEXTO:c SLASH R_BOTON CIERRA_ETIQUETA

|ABRE_HS list_hs:a R_CIERRAHS

list_hs::= list_hs:a op_hs

| op_hs

op_hs::= VAR1 VARIABLE:a IGUAL block_echo:b PUNTOYCOMA

| block_if

|R_ECHO block_echo:a PUNTOYCOMA

|block_repetir

| VAR2 VARIABLE:a IGUAL var_obj:b PUNTOYCOMA

| st_set

| st_insert

var_obj::= R_CREARPARRAFO ABRE lst_parrafo:a CIERRA

| R_CREARTEXTOA ABRE cad_var:a CIERRA

| R_CREARTEXTOB ABRE cad_var:a CIERRA

|R_CREARIMAGEN ABRE lst_img:a CIERRA

|R_CREARTABLA ABRE lst_table:a CIERRA

|R_CREARBOTON ABRE lst_boton:a CIERRA

st_set::= VAR2 VARIABLE:a CONCATENACION R_SETALTO ABRE expr_logica:b CIERRA PUNTOYCOMA

|VAR2 VARIABLE:a CONCATENACION R_SETPATH ABRE block_echo:b CIERRA PUNTOYCOMA

|VAR2 VARIABLE:a CONCATENACION R_SETANCHO ABRE expr_logica:b CIERRA PUNTOYCOMA

|VAR2 VARIABLE:a CONCATENACION R_SETBORDE ABRE expr_logica:b CIERRA PUNTOYCOMA

|VAR2 VARIABLE:a CONCATENACION R_SETTEXTO ABRE block_echo:b CIERRA PUNTOYCOMA

|VAR2 VARIABLE:a CONCATENACION R_SETCONTENIDO ABRE block_echo:b CIERRA PUNTOYCOMA

|VAR2 VARIABLE:a CONCATENACION R_SETALINEACION ABRE block_echo:b CIERRA PUNTOYCOMA

st_insert::= VAR2 VARIABLE:a CONCATENACION R_INSERTAR ABRE CIERRA PUNTOYCOMA

| VAR2 VARIABLE:a CONCATENACION R_CLICKBOTON ABRE block_echo:b CIERRA PUNTOYCOMA

st_get::= VAR2 VARIABLE:a CONCATENACION R_GETALTO ABRE CIERRA

|VAR2 VARIABLE:a CONCATENACION R_GETPATH ABRE CIERRA

|VAR2 VARIABLE:a CONCATENACION R_GETANCHO ABRE CIERRA

|VAR2 VARIABLE:a CONCATENACION R_GETTEXTO ABRE CIERRA

|VAR2 VARIABLE:a CONCATENACION R_GETCONTENIDO ABRE CIERRA

|VAR2 VARIABLE:a CONCATENACION R_GETALINEACION ABRE CIERRA

lst_parrafo::= cad_var

| cad_var:a COMA cad_var:b

lst_img::= cad_var

| cad_var:a COMA expr_logica

| cad_var:a COMA expr_logica:b COMA expr_logica

lst_boton::= cad_var

| cad_var:a COMA cad_var

lst_table::= ls_row

ls_row::= ls_row:a COMA CORCHETEIZQUIERDA ls_col:b CORCHETEDERECHA

| CORCHETEIZQUIERDA ls_col:a CORCHETEDERECHA

ls_col::= ls_col:a COMA block_echo

| block_echo

block_echo::= block_echo:a CONCATENACION e_echo

| e_echo

e_echo::= expr_logica

block_if::= R_IF ABRE expr_logica:a CIERRA LLAVEIZQUIERDA list_hsp:b LLAVEDERECHA

| R_IF ABRE expr_logica:a CIERRA LLAVEIZQUIERDA list_hsp:b LLAVEDERECHA R_ELSE LLAVEIZQUIERDA list_hsp:c LLAVEDERECHA

block_repetir::= R_REPETIR ABRE expr_rep:a CIERRA LLAVEIZQUIERDA list_hsp:b LLAVEDERECHA

expr_rep::=expr_rep:a MAS expr_rep

| expr_rep:a MENOS expr_rep

| expr_rep:a POR expr_rep

| expr_rep:a ENTRE expr_rep

| MENOS1 expr_rep

%prec MENOS1

| ENTERO

|VAR1 VARIABLE

|st_get:a

expr_logica::= expr_logica:a AND expr_logica

| expr_logica:a OR expr_logica

|NOT expr_logica

|expr

expr ::=expr:a IGUAL_LOGICO expr

|expr:a DESIGUAL expr

|expr:a MAYOR expr

|expr:a MAYOR_IGUAL expr

|expr:a MENOR expr

|expr:a MENOR_IGUAL expr

|expr:a MAS expr

| expr:a MENOS expr

| expr:a POR expr

| expr:a ENTRE expr

| MENOS expr

%prec MENOS1

| ENTERO

|VAR1 VARIABLE

|R_TRUE

|R_FALSE

|CADENA

|st_get

|ABRE expr_logica:a CIERRA

PUNTOS IMPORTANTES DEL CODIGO

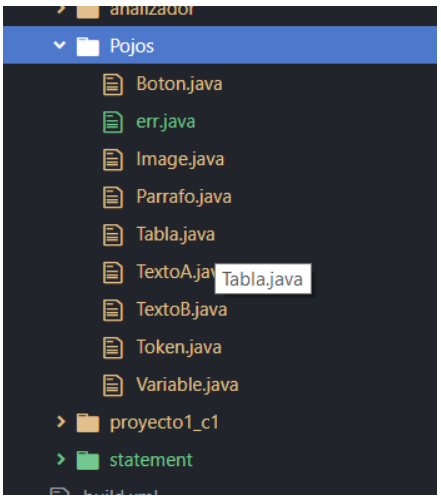
Se muestran algunos de los puntos importantes del código.

1. Listas globales estáticas, que guardan la lista de tokens, variables y la lista de errores, estas se encuentran en la clase Principal.

2. Método analizar, este método es el encargado de analizar todo el texto de entrada para su traducción a html, se encuentra en la clase Principal.

```
618
619     private void analizar(){
620         String input = txt_editor.getText();
621         lexer = new analizador.Lexer(new BufferedReader(new StringReader(input)));
622         parser = new analizador.parser(lexer,txt_console);
623         token.clear();
624         errorLexer.clear();
625         txt_console.setText("");
626         try {
627             parser.parse();
628             addTable();
629             generaHtml();
630             reporteTokens();
631             reporteLexer();
632             reporteSintactico();
633             /*for(Pojos.Token t : token){
634                 System.out.println(t.toString());
635             }*/
636         } catch (Exception ex) {
637             System.out.println(ex.getMessage());
638         }
639     }
```

3. Se crearon clases con el nombre de cada etiqueta que se pueda insertar, como por ejemplo Tabla, TextoA, etc. Para su manejo en el lenguaje Hscript.



4. Un punto muy importante en la aplicación, se centra en el package statement, que tiene una interface denominada st, esta obtiene todos los bloques recibidos por hscript como el bloque if, echo, repetir entre otros, cada uno de estos bloques tiene a su vez una lista de bloques st, esto con el fin de ir agregando los pedazos de bloques que va recibiendo el archivo, y al finalizar realizar las validaciones respectivas y si se cumplen ejecutarlos, la interface tiene un método denominado ejecutar, que es al que se manda a llamar en caso de que se cumpla la condición. Un ejemplo sobre esto se muestra a continuación.

```
    @Override
    public String ejecutar(javax.swing.JTextArea txt_console) {
        StringBuilder cadena = new StringBuilder();
        listst.stream().forEach((st) -> {
            if(st instanceof Var){
                cadena.append(st.ejecutar(txt_console));
            }
            else if(st instanceof Echo){
                cadena.append(st.ejecutar(txt_console));
            }
            else if(st instanceof St_If){
                if(((St_If)st).isCondicion())
                    cadena.append(st.ejecutar(txt_console));
            }
            else if(st instanceof St_IfElse){
                if(((St_If)st).isCondicion())
                    cadena.append(st.ejecutar(txt_console));
            }
            else if(st instanceof St_While){
                for(int i=0; i<((St_While)st).getRep();i++){
                    cadena.append(st.ejecutar(txt_console));
                }
            }
            else if(st instanceof Insert){
                cadena.append(st.ejecutar(txt_console));
            }
        });
        return cadena.toString();
    }

}
```

Bloque if.

```
    @Override
    public String ejecutar(javax.swing.JTextArea txt_console) {
        StringBuilder cadena = new StringBuilder();
        listst.stream().forEach((st) -> {
            if(st instanceof Var){
                cadena.append(st.ejecutar(txt_console));
            }
            else if(st instanceof Echo){
                cadena.append(st.ejecutar(txt_console));
            }
            else if(st instanceof St_If){
                if(((St_If)st).isCondicion())
                    cadena.append(st.ejecutar(txt_console));
            }
            else if(st instanceof St_While){
                for(int i=0; i<((St_While)st).getRep();i++){
                    cadena.append(st.ejecutar(txt_console));
                }
            }
            else if(st instanceof Insert){
                cadena.append(st.ejecutar(txt_console));
            }
        });
        return cadena.toString();
    }

}
```

Bloque repetir.

```
    @Override
    public String ejecutar(javax.swing.JTextArea txt_console) {
        txt_console.append("-> "+cadena + "\n");
        return "";
    }

}
```

Bloque echo.

```java
@Override
public String ejecutar(javax.swing.JTextArea txt_console) {
    lista_variable.put(vl.getId(),vl);
    return "";
}
```

Bloque variable.

5. Ejecución de las cadenas a través de Cup, posteriormente como lo mencionado arriba, los bloques se van llamando dentro de si, siempre y cuando se vayan cumpliendo las condiciones.