

# Instituto Tecnológico de Costa Rica



## **Administración en Tecnología de la Información**

### **Lenguajes de Programación**

**Prof. Andrei Fuentes**

### **Tarea programada #3**

### **Analizador de código SML**

José Rolando Li Acuña – 201220912

Verónica María Vargas Mora – 201269405

## Contenido

<b>Descripción del problema .....</b>	<b>3</b>
<b>Diseño del programa .....</b>	<b>4</b>
<b>Librerías utilizadas .....</b>	<b>10</b>
<b>Análisis de resultados .....</b>	<b>¡Error! Marcador no definido.</b>
<b>Lecciones aprendidas.....</b>	<b>17</b>

## Resumen ejecutivo

El presente documento consiste en la documentación de la aplicación web Analizador SML. La cual consiste en leer un archivo de texto que contiene código SML y muestra los resultados si ese código se ejecutara en el lenguaje de programación SML.

La aplicación nace de la necesidad de comprender con mayor facilidad las instrucciones de un código SML y mostrar el alcance que tiene cada variable según donde se encuentre.

Es importante aclarar que este analizador no es capaz de comprender todas las funcionalidades y expresiones permitidas en un código SML, únicamente podrá obtener el resultado de las siguientes expresiones:

- Asociaciones de variables.
- Expresiones let.
- Expresiones if-then-else.
- Valores.

Analizador SML, es una aplicación web sencilla y con poco pasos a ejecutar por el usuario pues lo único que se solicita es seleccionar desde un explorador de archivos, el archivo que se encuentre en su computadora y contenga código SML.

Luego de que el usuario lo seleccione la aplicación le mostrará las ocurrencias en el ambiente estático y ambiente dinámico representadas en dos tablas

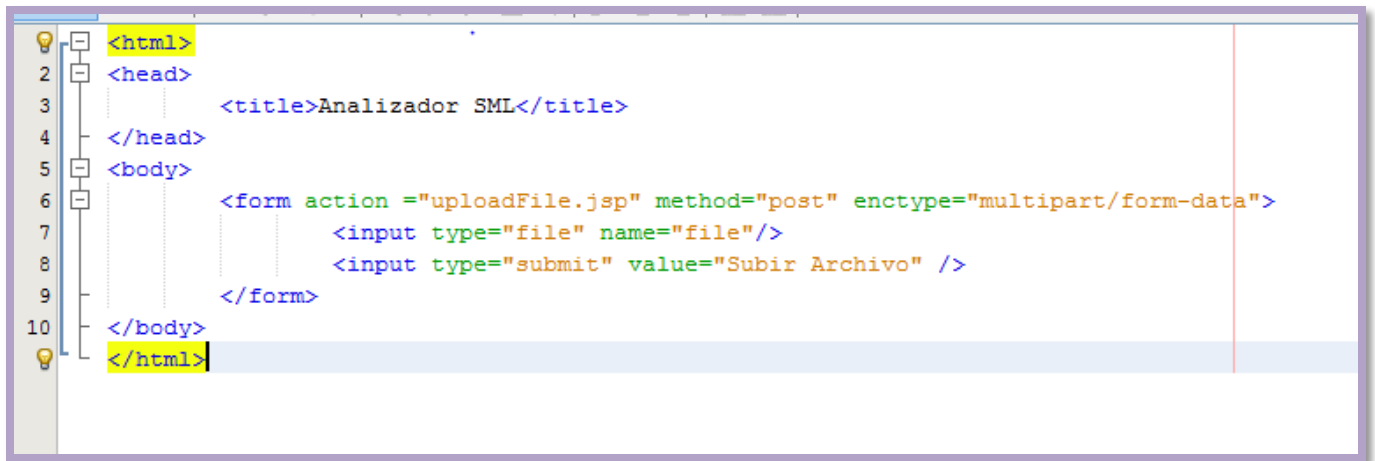
- **Tabla de ambiente estático:**  
Almacena las variables y el tipo de dato al cual pertenece la variable.
- **Tabla de ambiente dinámico:**  
Asocia el nombre de la variable a su valor exacto como expresión.

El lenguaje utilizado para dicha aplicación fue Java; complementado con jsp para el manejo de la web.

## Diseño del programa

### Aplicación web.

Para mostrar la aplicación web se utilizó la herramienta jsp. En primer lugar se diseña una página sencilla en html que muestra el botón para seleccionar el archivo desde un explorador de archivos y otro botón para enviar a analizar el código del archivo seleccionado.



```
<html>
<head>
  <title>Analizador SML</title>
</head>
<body>
  <form action = "uploadFile.jsp" method="post" enctype="multipart/form-data">
    <input type="file" name="file"/>
    <input type="submit" value="Subir Archivo" />
  </form>
</body>
</html>
```

Action = "uploadFile" es la instrucción que le indica a la aplicación web que hacer luego de cargar correctamente el archivo.

Una vez cargado el archivo correctamente, nos dirigimos al archivo uploadFile.jsp, donde se obtiene el archivo seleccionado, luego se almacena en una carpeta file ubicada en C:/file, para obtener la ruta completa y poder enviar esta al analizador de código.

```
8 <%
9
10 String ubicacion="C:/files/";
11 DiskFileItemFactory fac = new DiskFileItemFactory();
12 fac.setSizeThreshold(1024);
13 fac.setRepository(new File(ubicacion));
14
15 ServletFileUpload upd = new ServletFileUpload(fac);
16 String filePath = "";
17
18 try{
19     List<FileItem> partes = upd.parseRequest(request);
20     for (FileItem item : partes){
21         File file = new File(ubicacion,item.getName());
22         item.write(file);
23         filePath = ubicacion + item.getName();
24     }
25     SML s = new SML();
26
27
```

filePath: es la variable que almacena la ruta del archivo seleccionado para analizar.

Luego de esto se llama a la clase SML, la cual contiene todo el código respectivo para lograr el análisis correcto del archivo.

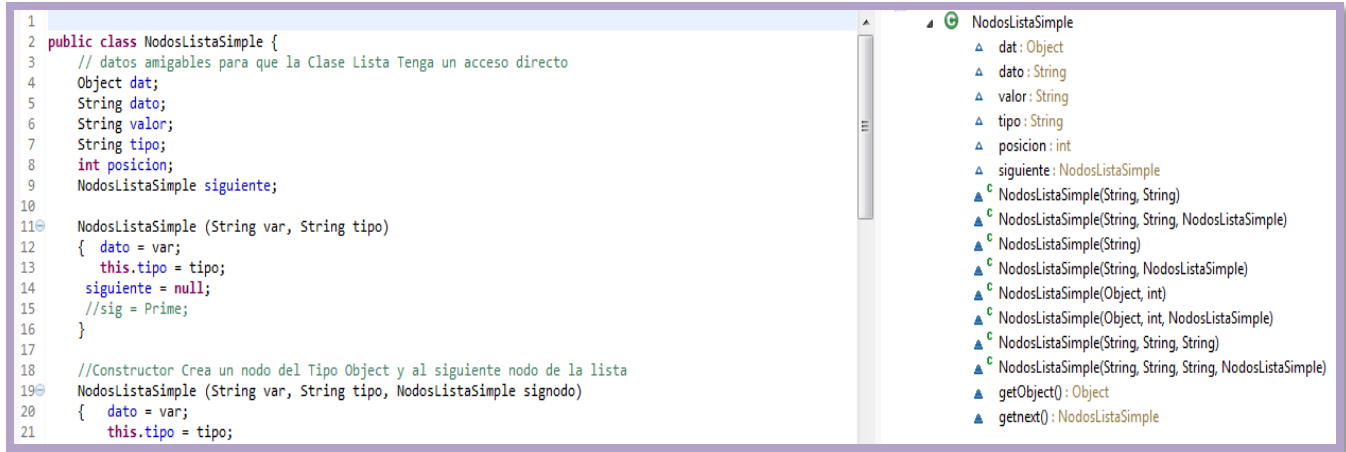
```
36
37 out.write("<br>");
38
39 SML s = new SML();
40 String resul = s.iniciar(filePath);
41 out.write(resul);
42
```

resul: es la variable que contendrá la información del ambiente dinámico y estático del código SML, y de inmediato lo muestra en pantalla.

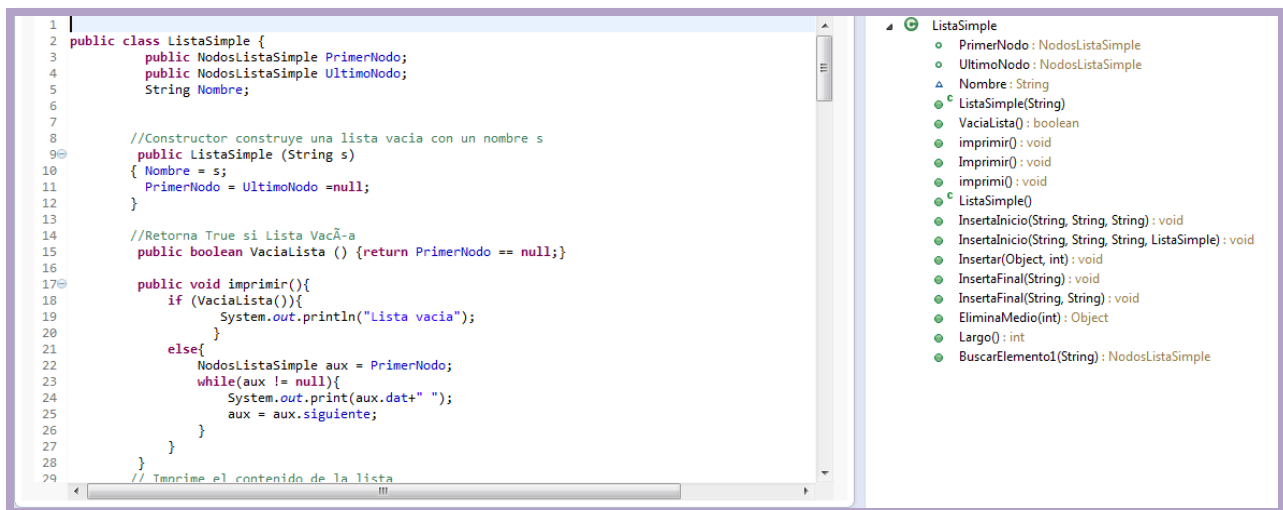
## Lógica – analizador de código

Para lograr un mejor manejo tanto del archivo como de los datos encontrados en este se implementa una estructura de datos propia, llamada ListaSimple, la cual a su vez requiere de nodos los cuales representan a cada elemento de la lista.

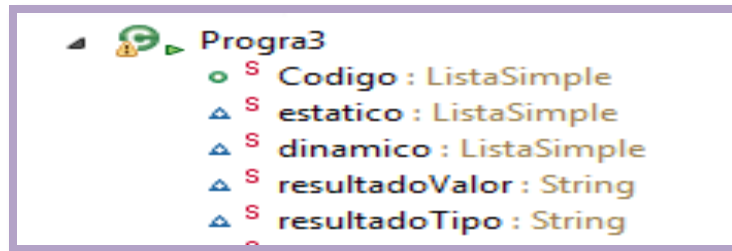
Descrita la importancia se presenta la clase NodoListaSimple en la siguiente imagen; destacan los constructores dependientes de la ariedad del nodo.



La estructura de lista simple destacan los diversos métodos de inserción: métodos para agregar nodos de maneras especifica; según las necesidades del programa para ubicar datos. El uso principal de la clase es almacenar el contenido del archivo sml.



Ya directamente en la solución se crean variables de clase en las que: se guarda el código leído, se guarda el ambiente estático y dinámico y se determina el valor y tipo de expresiones. Así se tiene respectivamente:



Con estas guías se lee el archivo y se analizar.

```

String ruta;
Scanner i = new Scanner(System.in);
System.out.println("");
System.out.print("Indique el nombre del archivo: ");
ruta = i.next();
FileReader fr = new FileReader(ruta);
BufferedReader br = new BufferedReader(fr);
String linea = br.readLine();
int pos=0;
while(linea !=null){
    StringTokenizer dato = new StringTokenizer(linea," ");
    while(dato.hasMoreTokens()){
        Object caracter = dato.nextToken();
        Codigo.Insertar(caracter, pos);
        pos++;
    }
    linea = br.readLine();
}
NodosListaSimple aux = Codigo.PrimerNodo;
analizar(aux,Codigo);
    
```

Como se nota en la última línea de la imagen anterior analizar recibe un primer nodo sobre el cual empezar a iterar y la lista en la que se encuentra el código sml del archivo. Esta analiza y determina el siguiente paso en el análisis según el código; ya sea definición con val, un let, un if u otra expresión.

43		
44	/*funcion analizar(aux,Codigo)	
47	public static void analizar(NodosListaSimple aux,ListaSimple Codigo) throws IOException{	
48	while(aux!=null){	
49	// Verifica si se define una variable	
50	if(aux.dat.equals("val")){	
115	//Si se presenta un let, llama a la funcion eval_let	
116	} else if(aux.dat.equals("let")){	
126	//Si se presenta un if, llama a la funcion arregloIf	
127	} else if(aux.dat.equals("if")){	
137	} else{	
162	}	
163	}	

Si se encuentra la palabra “val” se valida la siguiente palabra de ser un “let” llama a eval\_let; de ser un “if” llama a eval\_if; sino ejecuta:

```
else{
String valor = aux.siguiete.dat.toString();
estatico.InsertaFinal(variable);
dinamico.InsertaFinal(variable, aux.siguiete.dat.toString());
try{
//Verifica si el dato dado es un int
if (Integer.parseInt(valor) %1 == 0){
    NodosListaSimple aux2 = estatico.PrimerNodo;
    while(aux2!=null){
        if(aux2.dato.equals(variable)){ aux2.tipo = "int";break;}
        else aux2 = aux2.siguiete;
    }
    aux = aux.siguiete;
} //Fin del if
}
catch (NumberFormatException e){
//llama a la funcion verificaOper(), para determinar el tipo de dato dado
int band = verificaOper(valor);
if(band == 1);
else{
    NodosListaSimple aux2 = estatico.PrimerNodo;
    NodosListaSimple aux1 = dinamico.PrimerNodo;
    while(aux2!=null){
        if(aux2.dato.equals(variable)){ aux2.tipo = "int"; aux1 = dinamico.PrimerNodo; break;}
        else{ aux2 = aux2.siguiete;}
    }
    while(aux1!=null){
        if(aux1.dato.equals(variable)){ aux1.tipo = resultadoValor; break;}
        else aux1 = aux1.siguiete;
    }
    aux = aux.siguiete;
}
}
}
```

Guarda la variable y si logra convertirlo a int lo guarda como tipo entero; sino, llama a verificarOper().

Este método es eval\_if() recibe como parámetro un arreglo con todos los datos del if, la función lo que hace es ir recorriendo el arreglo separando por los datos o palabras claves que contiene un if que son “if”, “then” y “else”.

### Auxiliares de if

Arregloif(), evaluar() la primera para decidir el camino a tomar y la segunda para realizar las comparaciones.

El método eval\_then(): si es de largo 1 guarda lo que está dentro del “then”: si no verifica si es un “let” o un “if” por medio de las respectivas funciones.

Ahora se detalla eval\_let(): En si lo que hace es separar la lista en dos listas, variables y asignaciones. La primera guarda todo lo que este antes del “in” y la segunda lo que esta después del in hasta el “end”. Para después evaluar.

Tipo(): esta función determina el tipo del valor recién agregado; int o boolean.



```

496 public static void tipo(){
497     NodosListaSimple aux1 = dinamico.PrimerNodo;
498     NodosListaSimple aux2 = estatico.PrimerNodo;
499     while (aux1!=null && aux2!=null){
500         if (aux1.tipo.equals("true") || aux1.tipo.equals("false")){
501             aux2.tipo = "Boolean";
502             aux2 = aux2.siguiente;
503             aux1 = aux1.siguiente;}
504         else{
505             try{
506                 if (Integer.parseInt(aux1.tipo) %1 == 0){
507                     aux2.tipo = "int";
508                     aux2 = aux2.siguiente;
509                     aux1 = aux1.siguiente;}
510             }
511             catch (NumberFormatException e){
512                 aux2.tipo = define(aux1.tipo);
513                 aux2 = aux2.siguiente;
514                 aux1 = aux1.siguiente;}
515             }

```

VerificaOper(): crea una lista con las operaciones y llama a calcular()

Calcular(): prepara los datos para llamar a Operaciones() determinando el tipo de datos de las variables.

Operaciones(), Resolver(): son las que convierten a enteros o unifican los valores de haber variables; y determinan el resultado.

La función letra() : identifica el valor de las variables de ser necesario para operar con ellas.

```

629 public static int letra (String info) throws IOException{
630     if (dinamico.VaciaLista()){
631         System.out.println("Lista vacia");
632     }//Fin del if
633     else{
634         //Se realizan las búsquedas para realizar el cambio del valor en caso que sea neces
635         NodosListaSimple aux = dinamico.PrimerNodo;
636         NodosListaSimple guia = dinamico.PrimerNodo;
637         while (aux!=null){
638             String var = aux.dato;
639             if (var.equals(info)){
640                 while (guia != null){
641                     if (guia.tipo.equals(info)){
642                         guia.tipo = aux.tipo;
643                         break;}
644                     else{
645                         guia = guia.siguiente;}
646                 }
647             aux.tipo = aux.tipo;
648             break;}
649             else{
650                 aux = aux.siguiente;}
651         }
652         if(aux==null){resultadoTipo = define(info);
653             resultadoValor = info;

```

Define(): se utiliza más que todo para el ambiente estático puesto que lo que se hace en la función es identificar el tipo de dato de la variable. En esta parte se valida si es "int", "boolean", "char" o "tupla".

## Librerías utilizadas

- **java.io.BufferedReader:**  
Se necesita el almacenamiento en buffers de los caracteres obtenidos del archivo. El tamaño de la memoria intermedia con valor por defecto es suficiente para la mayoría de los propósitos de gran tamaño.
- **java.io.File:**  
Permite representación abstracta de las rutas de acceso de archivos y directorios.
- **java.io.FileNotFoundException:**  
Permite el manejo de errores si no se encuentra el archivo esperado dentro de la carpeta o ruta especificada en el código.
- **java.io.FileReader:**  
Librería para la lectura de archivos.
- **java.io.FileWriter:**  
Librería para la escritura de archivos.
- **java.io.IOException:**  
Excepciones producidas por operaciones fallidas o interrumpido de Entrada/ Salida.
- **java.io.PrintWriter:**  
Esta clase implementa todos los métodos de impresión que se encuentran en `PrintStream`.
- **java.util.ArrayList:**  
Permite la implementación de todas las operaciones de listas opcionales y permite todos los elementos, incluidos los nulos.
- **java.util.Iterator:**  
Iterator toma el lugar de empadronamiento en el marco de las colecciones de Java.
- **java.util.StringTokenizer:**  
La clase tokenizer permite una aplicación para romper una cadena en tokens.

# Manual de usuario

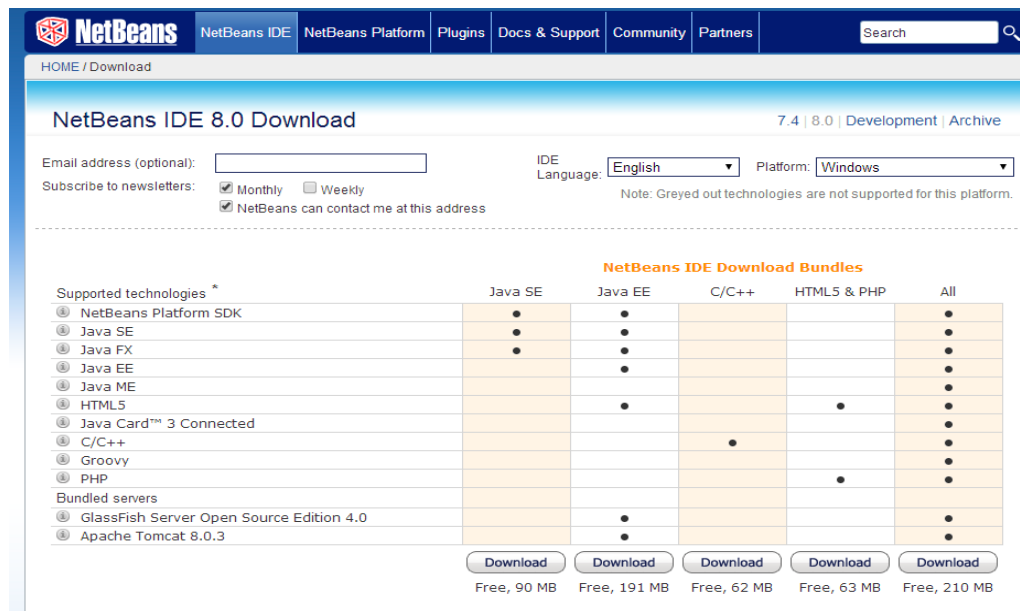
## Requerimientos

### Instalación y configuración de NetBeans IDE 8.0

1. Ingresamos a <https://netbeans.org/> La cual se muestra de la siguiente manera.



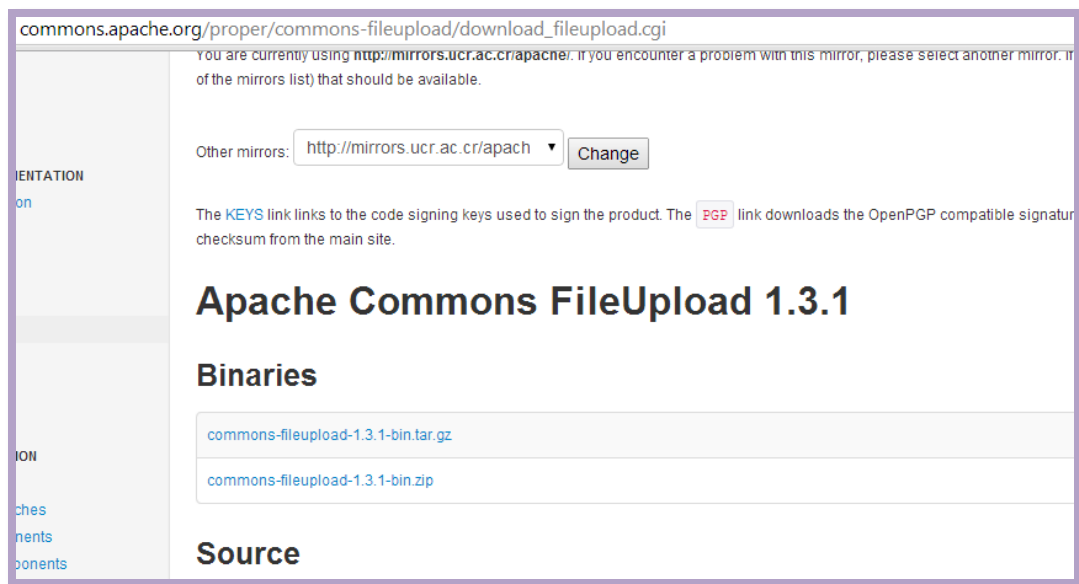
2. Nos dirigimos al botón naranja que dice Download, que se encuentra al lado derecho de la pantalla.



3. Le damos el botón que dice Download, debajo de Java EE. Luego de ello se inicia la descarga de NetBreans en nuestra computadora. Esperamos hasta que se descargue.
4. Una vez descargado por completo el NetBreans, le damos doble click y nos mostrara una pantalla, la cual le damos a todo siguiente hasta instalar por completo.

*Descargar librerías para aplicación web.*

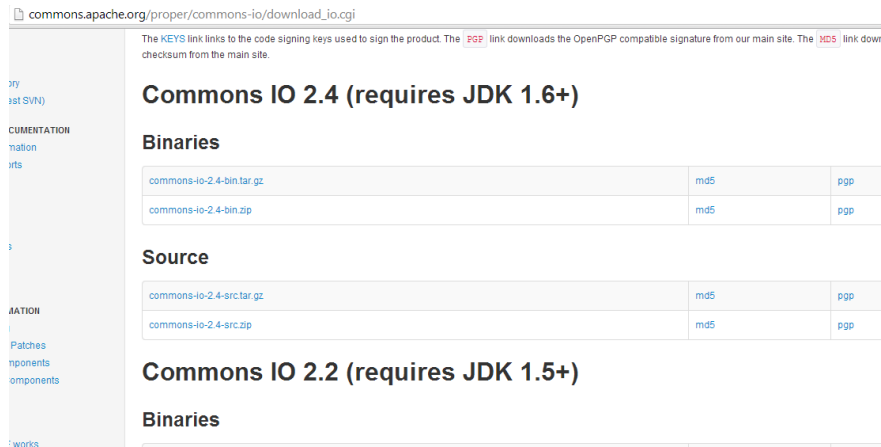
- Apache Commons FileUpload 1.3.1
1. Ingresamos a [http://commons.apache.org/proper/commons-fileupload/download\\_fileupload.cgi](http://commons.apache.org/proper/commons-fileupload/download_fileupload.cgi)



2. Damos click en commos-fileupload-1.3.1-bin-zip e inicia la descarga.

- Commons IO 2.4

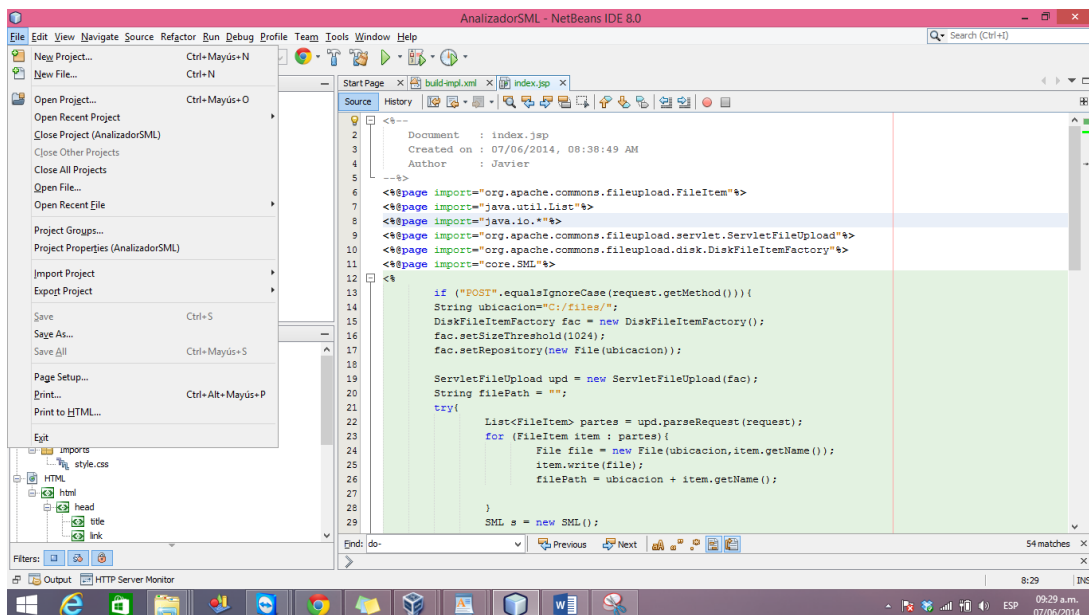
1. Ingresamos a [http://commons.apache.org/proper/commons-io/download\\_io.cgi](http://commons.apache.org/proper/commons-io/download_io.cgi)



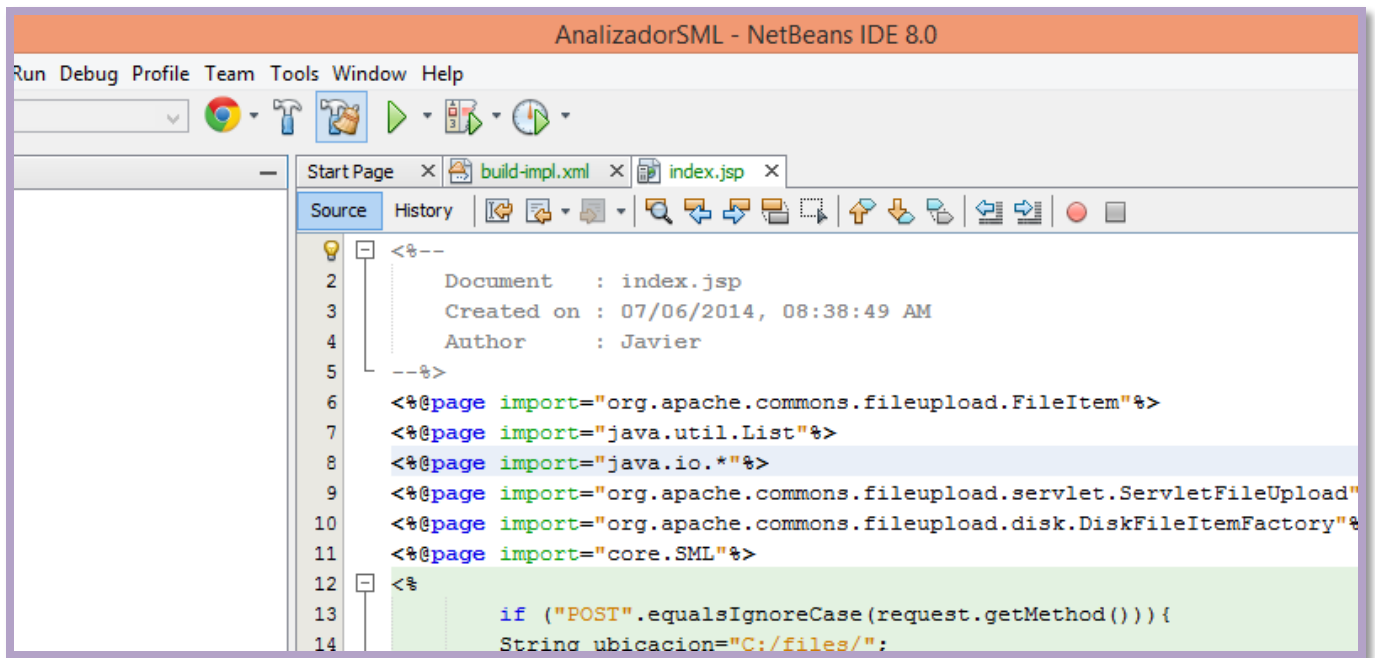
2. Damos click en [commons-io-2-4-bin-zip](#) e inicia la descarga.

## Ejecutar el programa

1. Descargamos el archivo de la aplicación.
2. Ingresamos a Netbeans y le damos en el menú File – Open Project



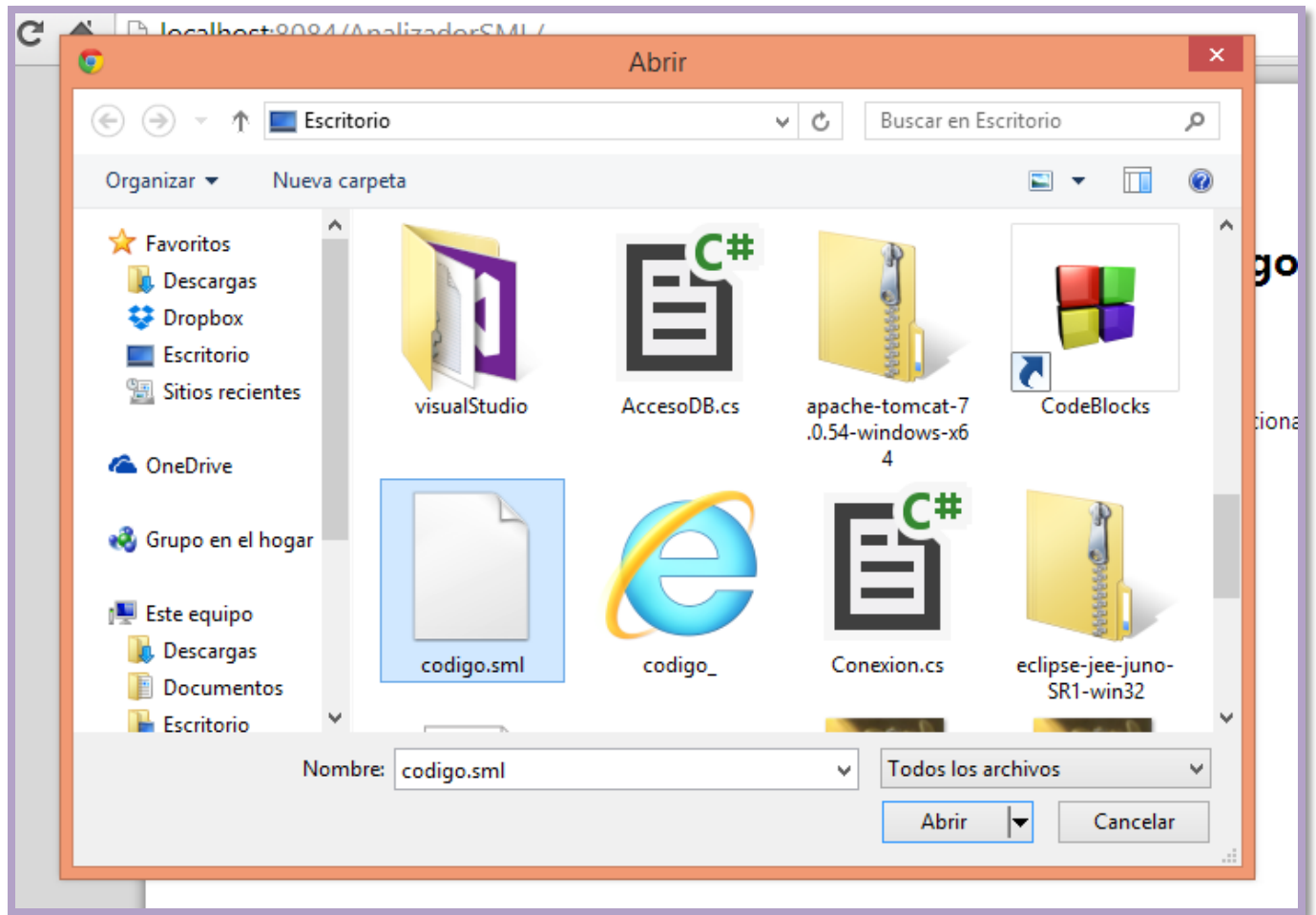
3. Seleccionamos el archivo de proyecto que descargamos.
4. Le damos click al botón verde para ejecutar el programa.

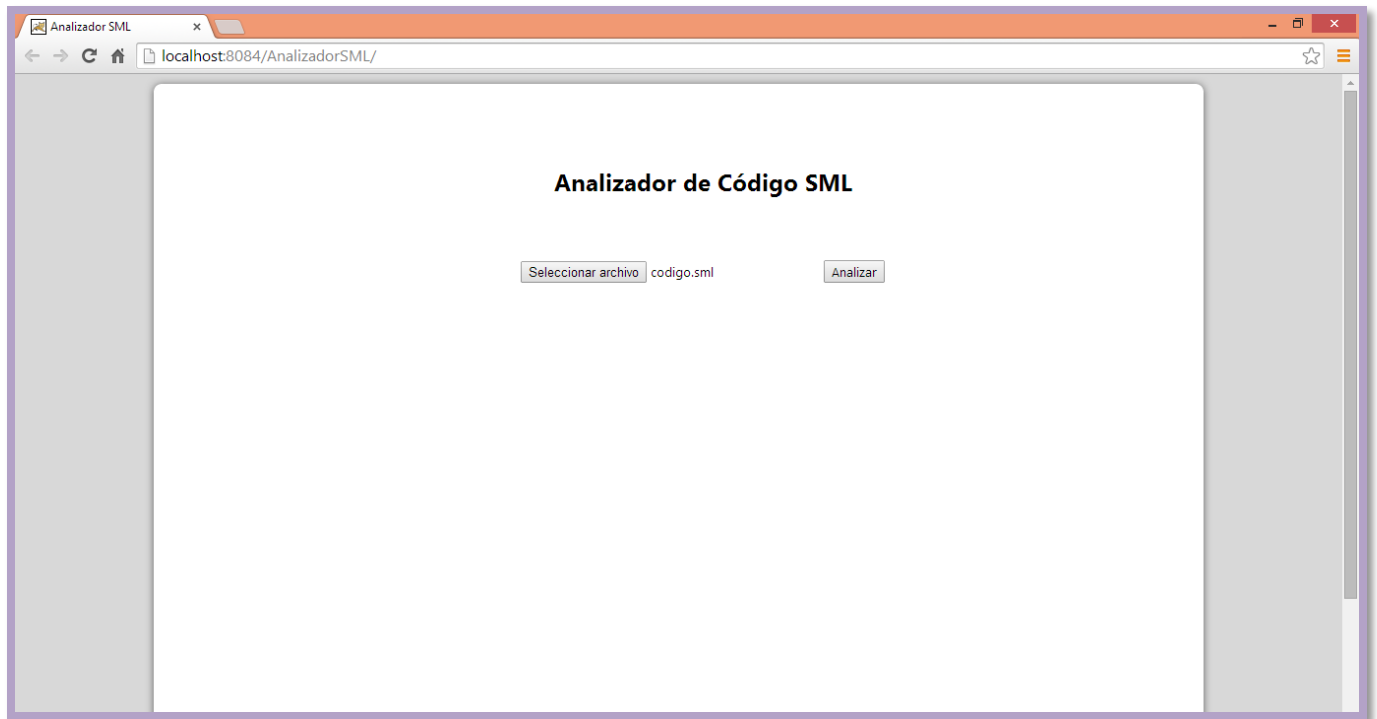


5. Una vez ejecutado el programa nos mostrará la pagina web.

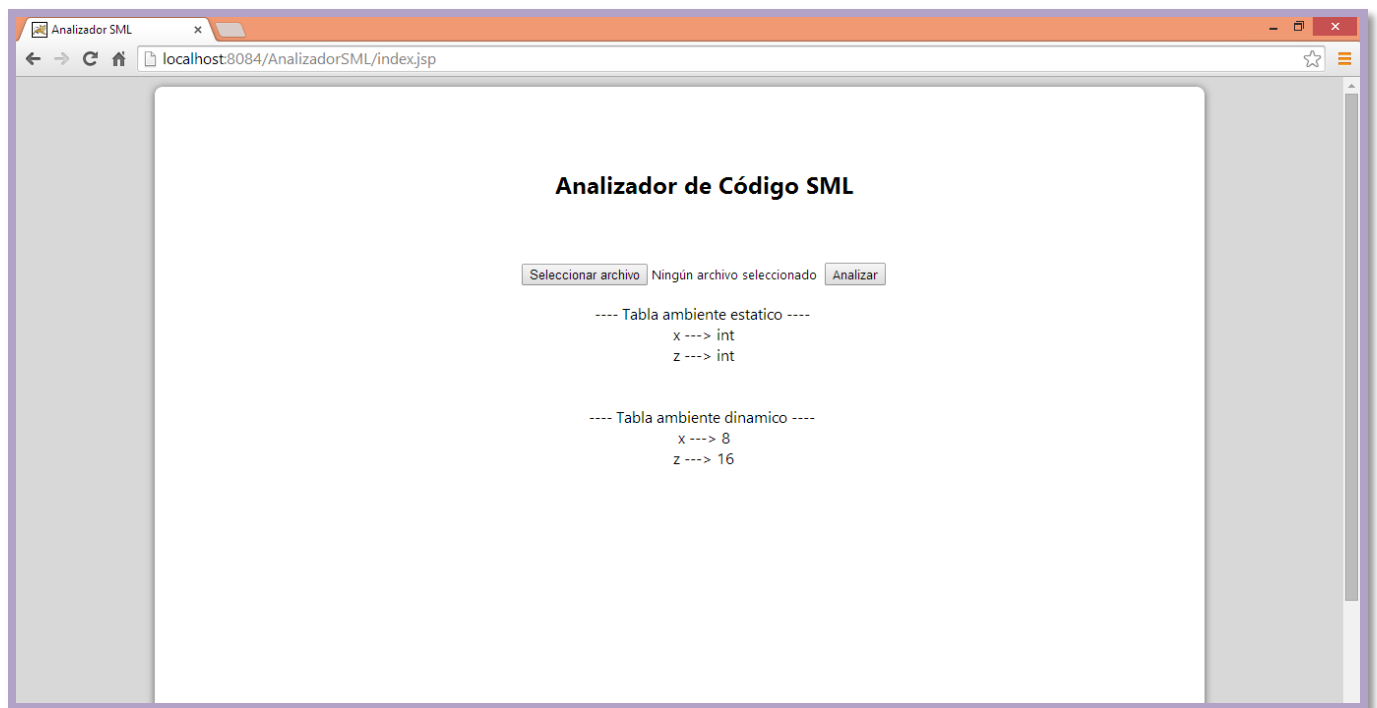


6. Damos click sobre el botón Seleccionar archivo, y buscamos el archivo que contiene el código SML el cual deseamos analizar.





7. Luego de seleccionado el archivo, le damos click al botón analizar. De inmediato nos mostrará el resultado al analizar el código SML





## Lecciones aprendidas

Como objetivo principal de este proyecto se expresan las perspectivas de los desarrolladores con respecto al proyecto.

### Jose Li

El desarrollo de esta aplicación web conlleva al desarrollo de habilidades para el trabajo en equipo e investigación; además mejoré mis conocimientos sobre el funcionamiento del lenguaje SML, y el manejo de los datos en los dos ambientes. Por otra parte aprendí sobre el diseño de aplicaciones web con el lenguaje html y java, su interacción y facilidades para futuras aplicaciones; y puse en práctica conceptos previos sobre java. Destacó también las características de la programación funcional dentro del proyecto.

### Verónica Vargas

Durante el desarrollo e implementación de esta tarea programada, la cual consiste en un analizador de ambiente estático y dinámico de un código en SML, desde el lenguaje de programación JAVA aprendí varios aspectos tanto del lenguaje SML como del paradigma funcional, las cuales son:

Logré tener una mejor claridad de como escribir códigos de SML, los alcances de cada expresión, variable o valor, también como se maneja el shadowing de las variables, y diferentes maneras para evaluar una expresión. Además, conocer más sobre el paradigma funcional, y algunas de las funcionalidades de este tipo de lenguaje, ya que, para lograr hacer el análisis en ambiente estático y dinámico tuvimos que investigar sobre dicho paradigma y más específicamente sobre código SML.”