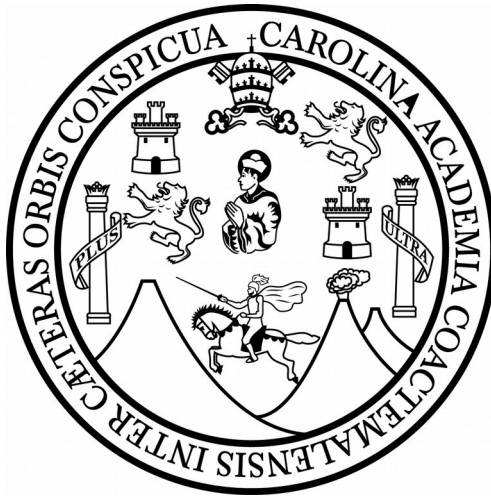


CENTRO UNIVERSITARIO DE OCCIDENTE

DIVISIÓN CIENCIAS DE LA INGENIERÍA

CARRERA DE INGENIERÍA CIENCIAS Y SISTEMAS



LABORATORIO DE ORGANIZACIÓN LENGUAJES Y COMPILADORES 2

“SEXTO SEMESTRE”

ING.: JOSÉ MOISÉS GRANADOS GUEVARA

AUX: EDVIN TEODORO GONZALEZ RAFAEL

ESTUDIANTE: Bryan René Gómez Gómez – 201730919

PROYECTO: “Manual Usuario – PRIMER PROYECTO - ANALIZADOR”

FECHA: 03 de Septiembre de 2,020

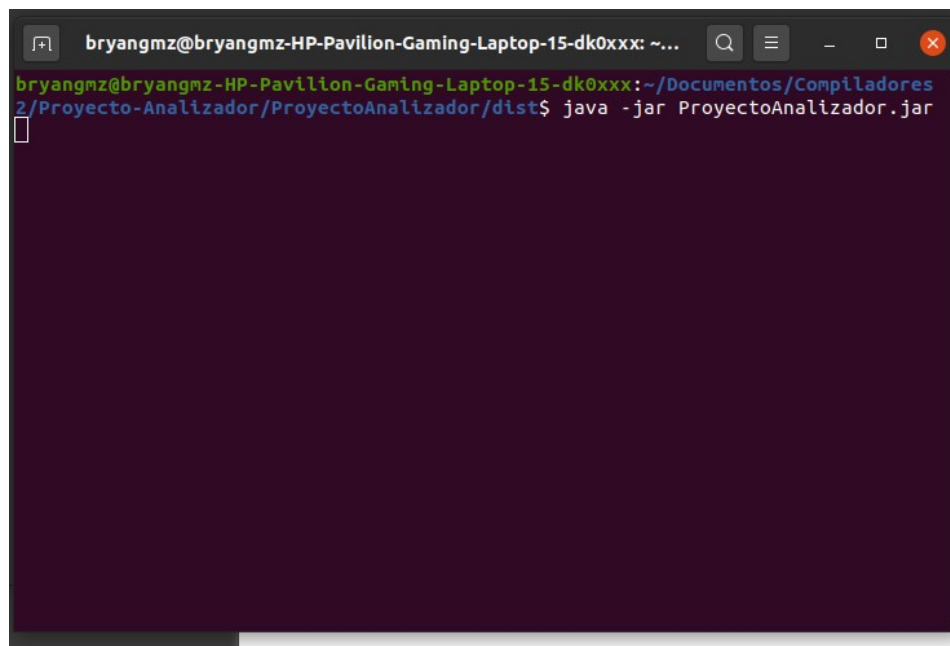
ANALIZADOR – GUATEMALA

La aplicación consiste en un editor de código que tenga la capacidad de reconocer y compilar una serie de lenguajes, para ello la aplicación deberá tener un repositorio en donde se almacenarán los archivos que contienen la definición de las expresiones regulares, gramática y reglas semánticas de los lenguajes soportados por la aplicación.

Para agregar un nuevo lenguaje al repositorio se hará a través de un archivo de entrada (.len) por medio de una opción del menú principal de la interfaz gráfica. Después de haber ingresado el nuevo lenguaje este deberá aparecer en el menú principal de la aplicación para que el usuario lo pueda seleccionar y usar cuando lo requiera.

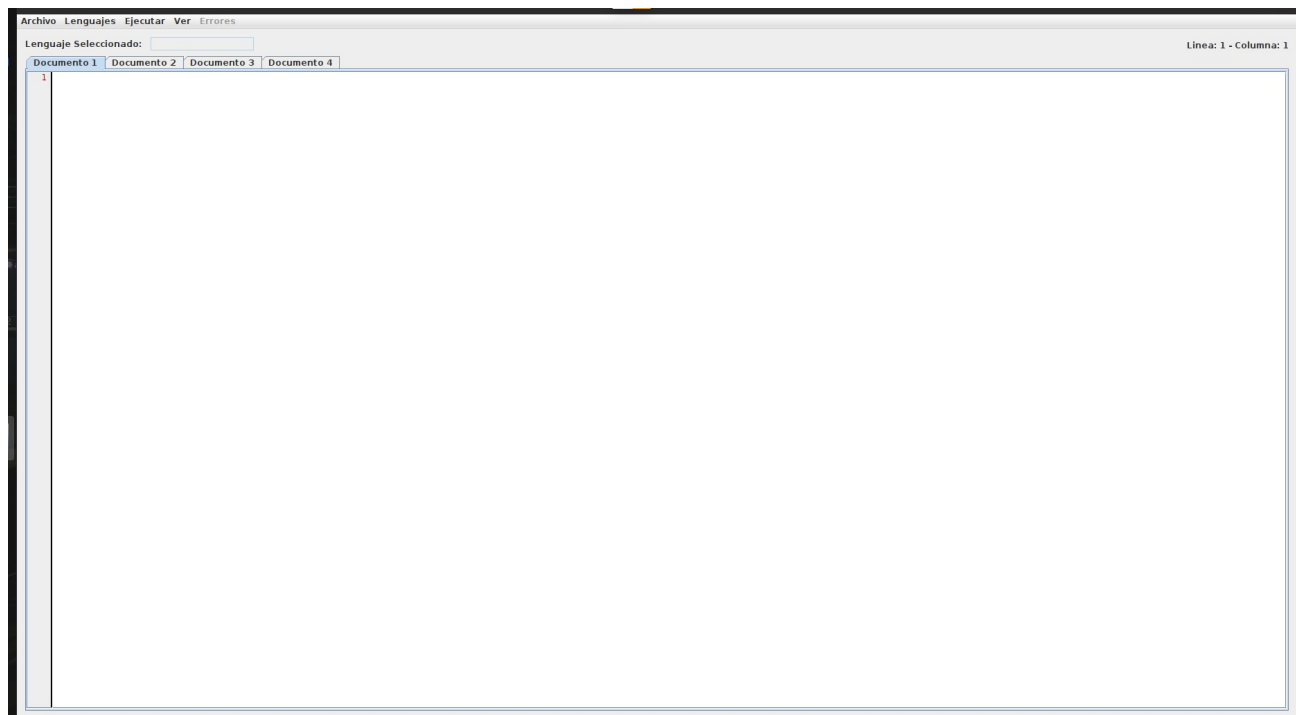
Para compilar una cadena escrita por el usuario, este deberá previamente seleccionar un lenguaje y la aplicación deberá generar un analizador de sintaxis ascendente (LALR) el cual tomará como base la gramática del lenguaje seleccionado que se encuentra almacenada en el repositorio, también se deberá tomar en cuenta que las gramáticas de los lenguajes tienen asociadas a sus producciones reglas semánticas las cuales producen resultados que también deberán ser mostrados por la aplicación.

Para iniciar la aplicación debes de tener instalado JDK 8: Debes de abrir la terminal en donde se encuentra el .jar ejecutable. Debes de ingresar el siguiente comando: **\$java -jar ProyectoAnalizador.jar**



```
bryangmz@bryangmz-HP-Pavilion-Gaming-Laptop-15-dk0xxx: ~...
bryangmz@bryangmz-HP-Pavilion-Gaming-Laptop-15-dk0xxx: ~/Documentos/Compiladores
2/Proyecto-Analizador/ProyectoAnalizador/dist$ java -jar ProyectoAnalizador.jar
```

Interfaz Grafica

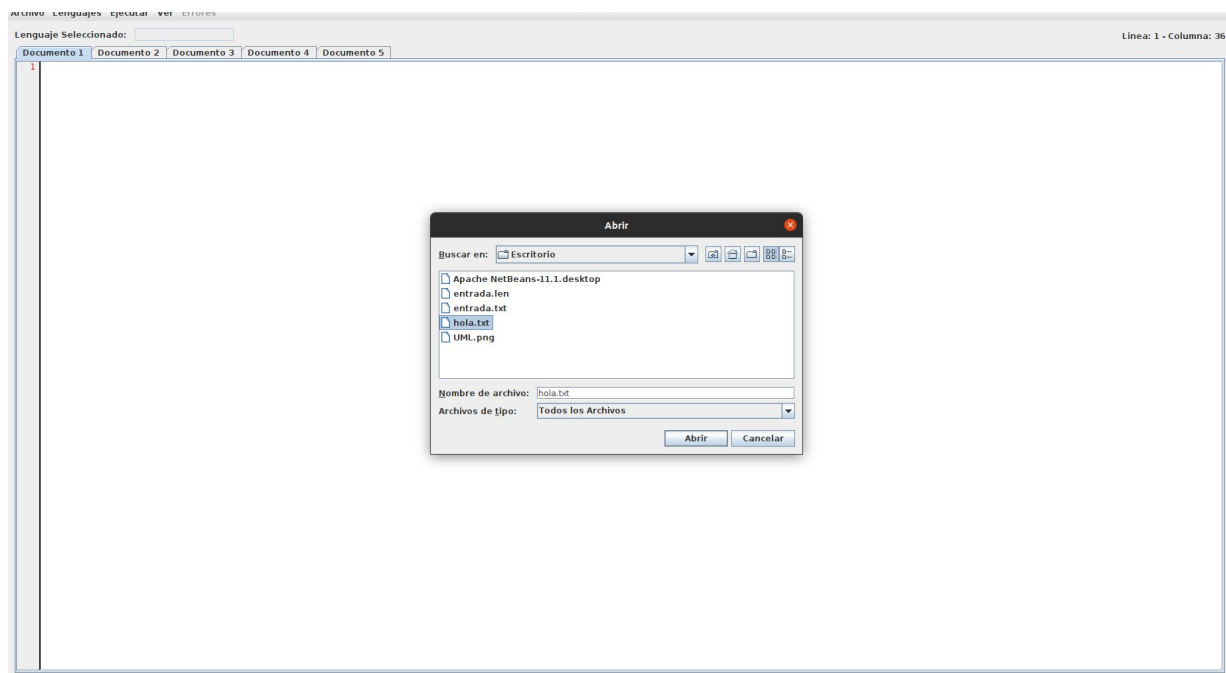


Menús

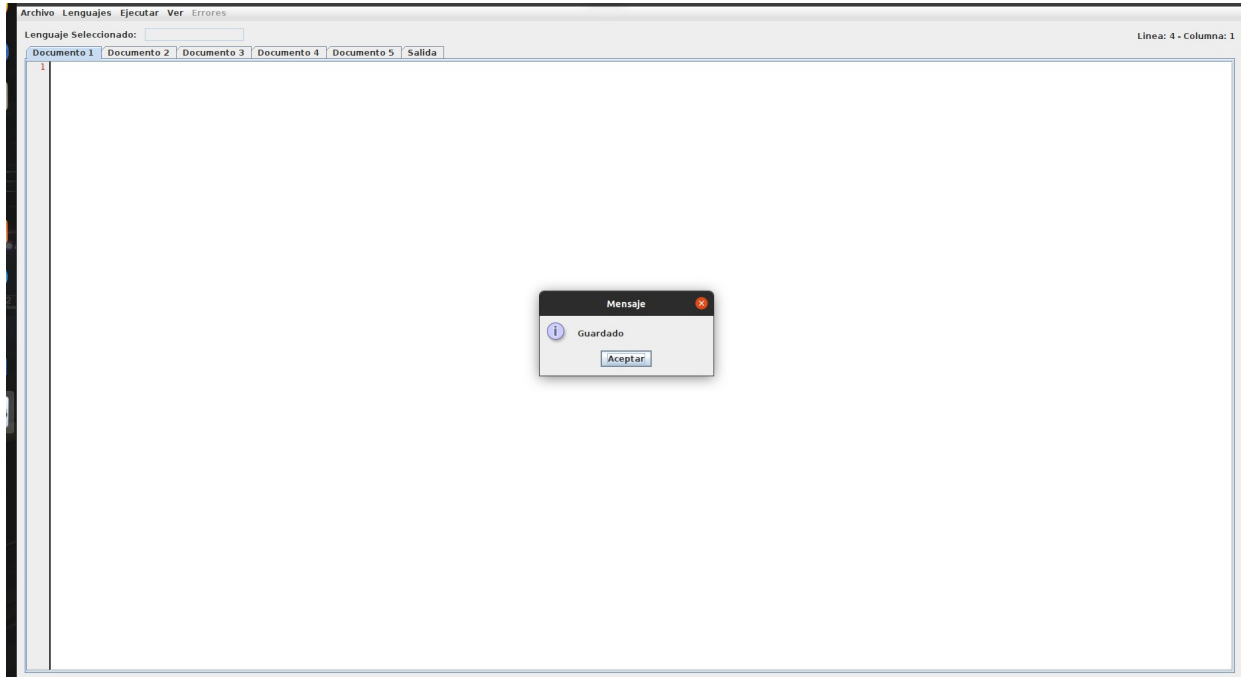
Cuenta con los siguientes submenús:

Archivo

- **Nuevo:** Despliega una nueva pestaña, en la cual el usuario podrá editar un archivo.
- **Abrir:** Permite abrir un archivo para que este pueda ser editado.



- **Guardar:** Permite guardar un archivo en la ubicación que el usuario desea, el archivo deberá de guardarse con la extensión del lenguaje seleccionado.
- **Guardar Como:** Esta opción permite guardar un archivo previamente almacenado con diferente nombre, el archivo deberá de guardarse con la extensión del lenguaje seleccionado.



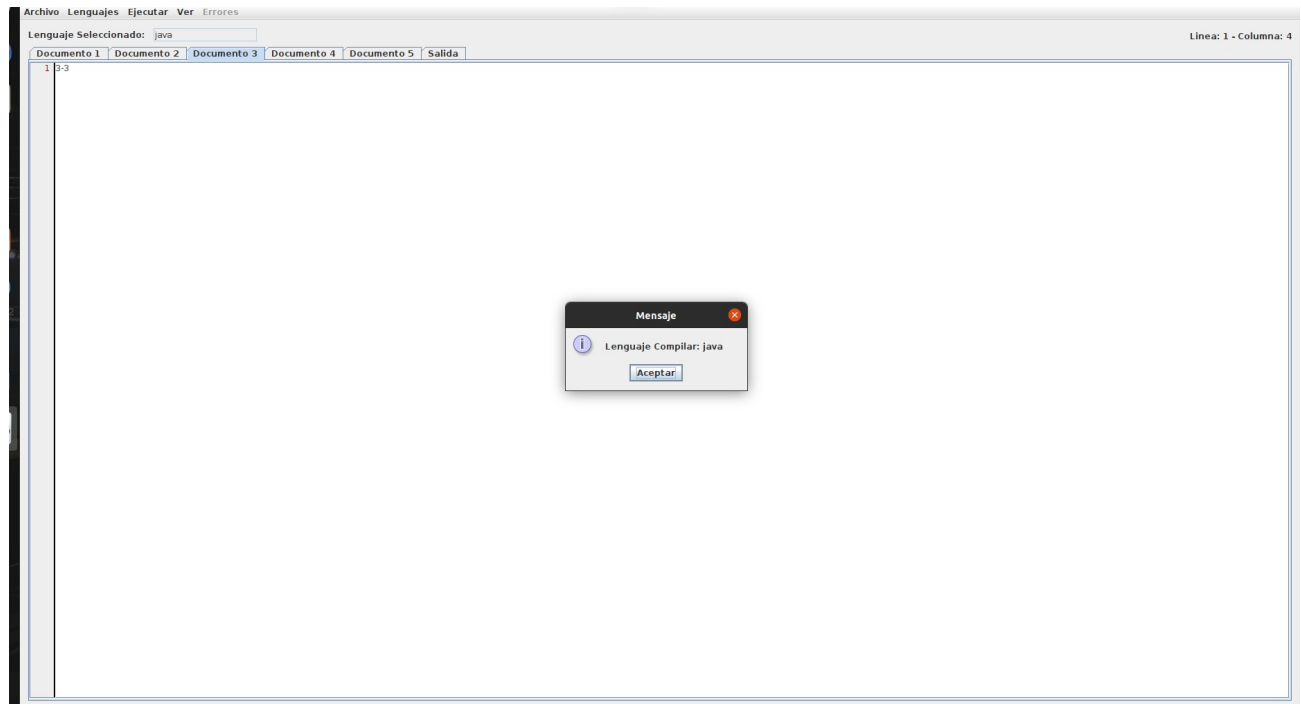
- **Salir:** Con esta opción se cierra la aplicación.

Lenguajes

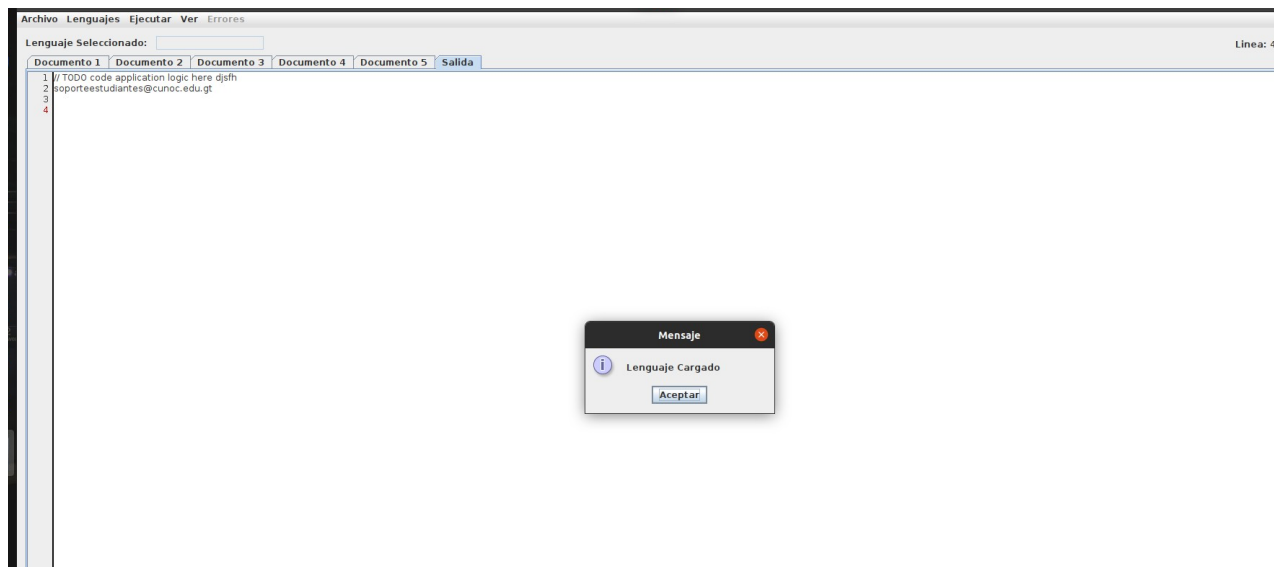
En este sub-menú se deben mostrar todos los lenguajes que la aplicación soporta y permite seleccionar el lenguaje con el que se va a trabajar.

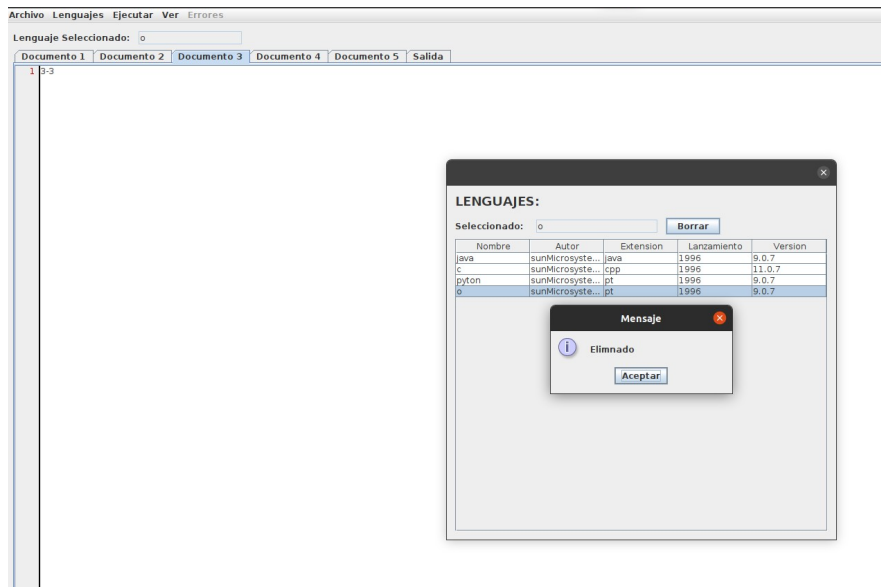
Ejecutar

- **Compilar:** A través de esta opción se podrán compilar la o las cadenas de entrada ingresadas por el usuario. Se deberá tomar en cuenta que previamente el usuario deberá seleccionar un lenguaje para compilar dichas cadenas. El contenido que se tomará en cuenta para realizar este proceso será el que esté en la pestaña activa.



- **Cargar Lenguaje:** Por medio de esta opción se podrán agregar nuevos lenguajes al repositorio de la aplicación, los cuales posteriormente deberán aparecer en el menú principal para que puedan ser seleccionados por el usuario para realizar el proceso de compilación. Se deberá tomar en cuenta que el lenguaje no podrá ser cargado a la aplicación sin estar libre de errores (léxicos, sintácticos y semánticos). El contenido que se tomará en cuenta para realizar este proceso será el que esté en la pestaña activa. Si ya existe una carga previa del mismo lenguaje, entonces se debe reemplazar por la carga más reciente.





- **Borrar Lenguaje:** Por medio de esta opción se podrá eliminar un lenguaje del repositorio de la aplicación el cual también deberá ser eliminado del menú principal. Para esto la aplicación deberá mostrar un cuadro de diálogo en el cual

deberá contener un combobox con todos los lenguajes que la aplicación soporta para que el usuario pueda seleccionar el que desea eliminar.

Ver

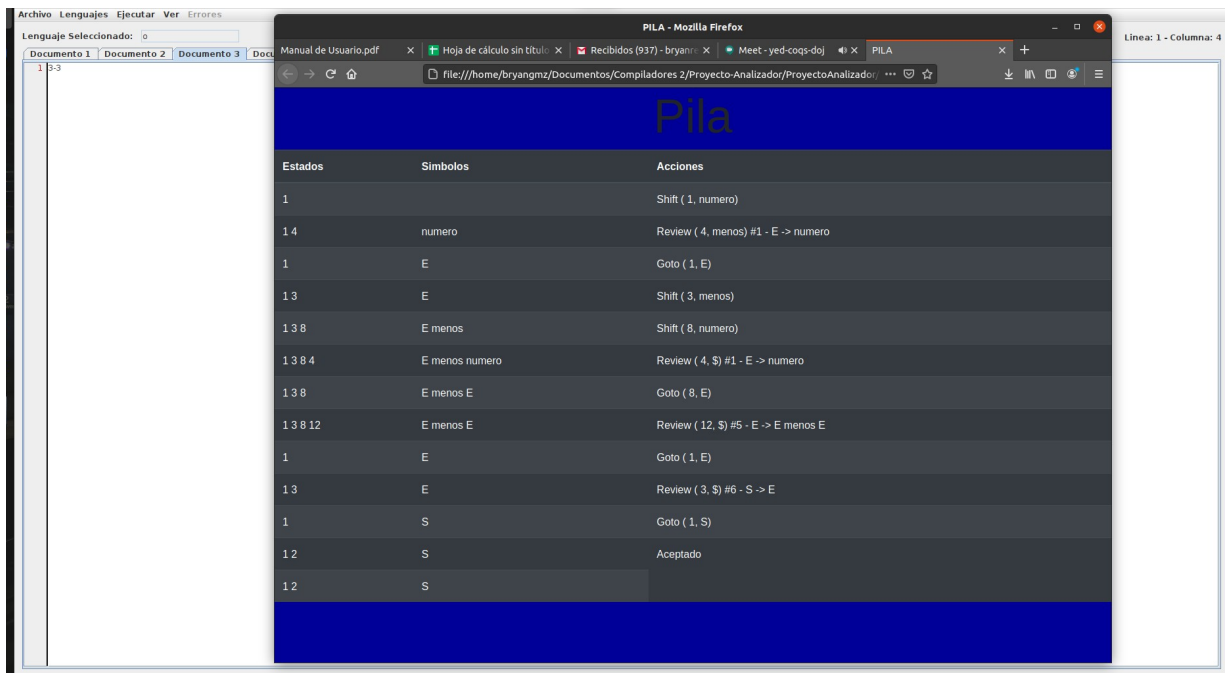
Esta opción se habilita después de que el usuario haya compilado una cadena de entrada con un determinado lenguaje, en caso contrario permanecerá bloqueada.

- **Tabla LALR:** Esta opción deberá desplegar la tabla LALR, la cual fue generada en el momento que el usuario seleccionó un lenguaje para compilar una cadena de entrada.

The screenshot shows the application's main window with the 'Tabla LALR' window open. The window displays a table with 13 rows and 12 columns, representing the LALR table for the selected language.

#Estado	div	por	menos	más	id	numero	pc	pa	\$	F	T	E
1	---	---	---	---	S7	S6	---	S5	---	G4	G3	G2
2	---	---	---	S8	---	---	---	---	A	---	---	---
3	---	---	---	R6	---	---	R6	---	R6	---	---	---
4	---	S9	---	R5	---	---	R5	---	R5	---	---	---
5	---	---	---	---	S7	S6	---	S5	---	G4	G3	G10
6	---	R2	---	R2	---	---	R2	---	R2	---	---	---
7	---	R3	---	R3	---	---	R3	---	R3	---	---	---
8	---	---	---	---	S7	S6	---	S5	---	G4	G11	---
9	---	---	---	---	S7	S6	---	S5	---	G4	G12	---
10	---	---	---	S8	---	---	S13	---	---	---	---	---
11	---	---	---	R7	---	---	R7	---	R7	---	---	---
12	---	---	---	R4	---	---	R4	---	R4	---	---	---
13	---	R1	---	R1	---	---	R1	---	R1	---	---	---

- **Pila:** Esta opción deberá desplegar en pantalla el contenido de la pila que se utilizó para reconocer una cadena de entrada previamente ingresada por el usuario. En caso contrario la aplicación deberá mostrar un mensaje indicando que no se ha reconocido ninguna cadena.



Sintaxis del archivo que define un lenguaje (Carga de un nuevo lenguaje SINTAXIS)

Estructura del archivo

La estructura del archivo para agregar un nuevo lenguaje, está compuesta por cinco secciones: información del lenguaje, código fuente, expresiones regulares, símbolos terminales y no terminales, finalmente se define la gramática.

Información Lenguaje

%%

Código fuente

%%

Expresiones Regulares

%%

Símbolos Terminales y No Terminales

%%

Gramática

Se puede realizar comentarios en cualquier sección del archivo usando de la siguiente forma:

- **Una línea:** Esto permite hacer un comentario en una línea.

// Este es un comentario

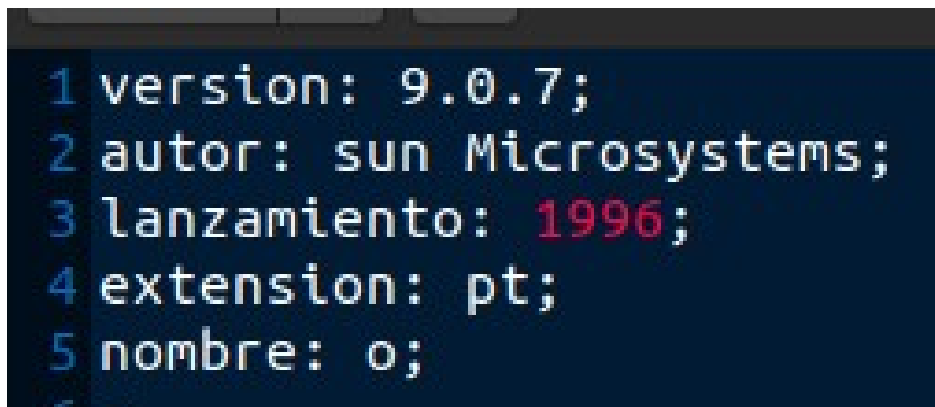
- **Varias líneas:** esto permite realizar comentarios en una o más líneas del archivo.

/* Este es un comentario */

Información Lenguaje

En esta sección se deberá de insertar información acerca del lenguaje que se está definiendo en el archivo, los campos de esta sección son los siguientes:

- Nombre: Nombre del lenguaje
- Versión: la versión del lenguaje
- Autor: el autor del lenguaje
- Lanzamiento: el año del lanzamiento del lenguaje.
- Extension: extensión de los archivos de cada lenguaje.

A screenshot of a code editor with a dark blue background. It displays five lines of code, each numbered from 1 to 5 in the left margin. The code is written in a light blue monospace font. Line 1: '1 version: 9.0.7;'. Line 2: '2 autor: sun Microsystems;'. Line 3: '3 lanzamiento: 1996;'. Line 4: '4 extension: pt;'. Line 5: '5 nombre: o;'. The text '1996' in line 3 is highlighted in red.

Esta información es crucial cuando se lee el archivo, ya que mediante estos campos se obtendrá el nombre del lenguaje y que posteriormente se podrá seleccionar identificándose con el nombre indicado en esta sección, se podrá consultar también la información sobre cada lenguaje y básicamente consiste en mostrar esta información al usuario.

El único campo obligatorio de esta sección es nombre, los demás datos son opcionales.

Sección de Código Fuente

En esta sección deberá ir el código fuente java de las funciones o procedimientos que serán utilizados en las reglas semántica. Se asume que este código está libre de errores por lo que no es necesario un análisis del mismo.

Nota: Esta sección es opcional, es decir puede o no venir en el archivo de entrada.


```
%% //codigo fuente
List<String> listString=new ArrayList<>();

public void addString(String valor){
    listString.add(valor);
}

public Integer suma(Integer numero1,Integer numero2){
    return numero1+numero2;
}
```

Sección de Expresiones Regulares

En esta sección deberán ser declaradas las expresiones regulares para establecer la forma que tendrá cada token que es válido para un determinado lenguaje.

La sintaxis de esta sección está compuesta por un identificador y la expresión regular que corresponde al identificador.

- **Identificador:** el identificador será el nombre que tendrá cada token que cumpla con la expresión regular especificada.
- **Expresión Regular:** Permite establecer la forma que tendrá cada token que es válido para el lenguaje que se está definiendo.

```
%% //Sección de expresiones regulares
palabra = [a-z]+;
numero = [0-9]+;
punto = ".";
decimal = [0-9]+((.)([0-9]+)?);
más = "+";
menos = "-";
por = "*";
& = [\n\t]; /* Significa que cuando se analice un archivo,
<identificador1> = <expresión regular1>;
<identificador2> = <expresión regular2>;
```

Se puede emplear el identificador “&” para indicar que la expresión regular deberá ser ignorada, cuando se analice un archivo.

Ejemplo:

& = [\n\t]; /* Significa que cuando se encuentre este token deberá ser ignorado */

Sección de símbolos terminales y no terminales

En esta sección se deberán declarar los símbolos terminales y no terminales del lenguaje que se esté definiendo.

```
9
0 terminal cadena por, div;
1 terminal real más, menos;
2 terminal entero numero, id;
3 terminal pa, pc;
4
5
6 no terminal S, E;
7
8
```

Sintaxis:

<no terminal | terminal> [entero|real|cadena] id1[,id2,.....,idn];

Nota: Los símbolos no terminales estarán escritos con letras MAYÚSCULAS y los terminales en letras minúsculas.

Nota: La precedencia de los símbolos terminales se asignará de acuerdo al orden en que se declaren y se tomará la asociatividad por la izquierda, por lo que no deberá haber conflictos en la tabla LALR. En el caso del ejemplo anterior los símbolos “por” y “div” tienen mayor precedencia que “mas” y “menos”.

Sección de gramática

En esta sección se definirá la gramática del lenguaje y las reglas semánticas asociadas a cada producción.

Las reglas semánticas van encerradas entre llaves al final de la producción y antes del punto y coma, y estas son opcionales para cada producción, este código se deberá de ejecutar al momento de reducir la producción. Cada símbolo No Terminal y Terminal puede tener una estructura o variable asociada del

tipo declarado previamente, la cual se declarará con “:nombre” y existirá un atributo RESULT asociado a cada lado izquierdo de las producciones.

Sintaxis:

no terminal :: [<terminal | no terminal>[:nombre]] [{reglas semánticas}];

Nota: Si se desea una producción con lambda (cadena vacía), se deberá dejar la producción del lado derecho solo con punto y coma.

Ejemplo:

E :: ;

Nota: El código fuente de las reglas semánticas debe estar escrito en java y se asume que este código está libre de todo tipo de errores, por lo cual se deberá omitir el análisis léxico, sintáctico y semántico, pero si al momento de ejecutarse este código se generan errores estos deberán ser mostrados en la pestaña de errores de la aplicación.

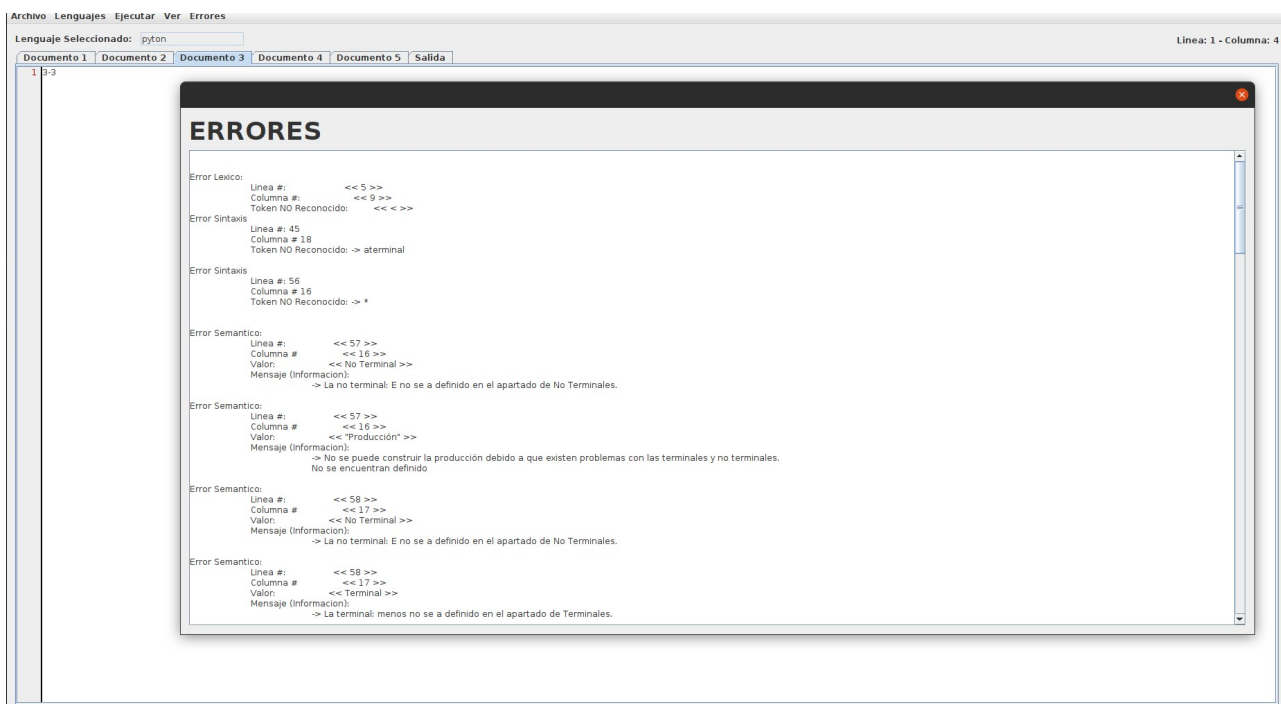
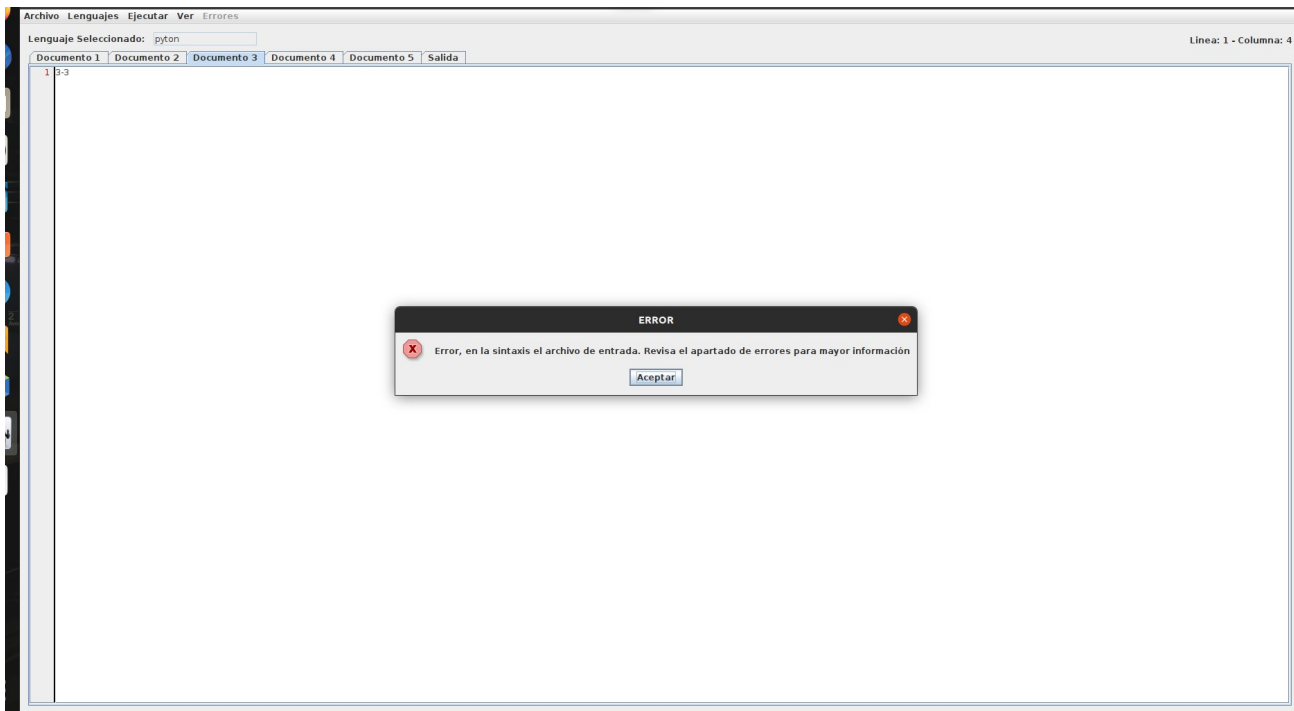
```
61
62 S :: E:val {printf("Resultado = %d",val);};
63 E :: E:val menos E:val2 {RESULT=val - val2;};
64 E :: E:val más E:val2 {RESULT=val + val2;};
65 E :: E:val por E:val2 {RESULT=val * val2;};
66 E :: E:val div E:val2 {RESULT=val / val2;};
67 E :: numero:val {RESULT=val;};
68
69
70
71 A :: numero X Y;
72 X :: más numero;
73 X :: numero;
74 X :: div numero;
75 X :: ;
76 Y :: menos numero;
77
78
79
80 /*
81 E :: E más T;
82 E :: T;
83 T :: F;
84 T :: F por T;
85 F :: id;
86 F :: numero;
87 F :: pa E pc;
88 */
89
90 /*
91
92 S :: A más E:val {printf("Resultado = %d",val);};
93 S :: E:val {RESULT=val - val2;};
94 E :: A;
95 A :: numero:val {RESULT=val;};
96 A :: por E:val2 {RESULT=val + val2;};*/
```

Manejo de errores

La aplicación deberá de reportar los errores en una tabla indicando:

- Tipo error: lexico, sintactico o semantico.
- fila: fila donde se encuentra el error.
- columna: columna donde se encuentra el error.
- token: el token que provocó el error.

Es importante considerar la recuperación de errores, ya que la aplicación deberá de reportar todos los errores que tenga el archivo de un lenguaje o el código fuente del editor.



REQUISITOS MÍNIMOS PARA ESTE PROGRAMA:

- Microsoft .NET Framework versión 4.0
- Windows XP, Vista, 7, 8, 8.1, 10
- Linux Ubuntu 15.04 y Superior:
- Espacio en disco: 100 MB de espacio libre en disco
- Pentium 1 GHz o superior con 1GB de RAM o más