



## Manual de Usuario



Manual de usuario para la herramienta didáctica GADUN  
Programa de Ingeniería de Sistemas  
Facultad de Ingeniería  
Universidad de Nariño  
2020

## INTRODUCCIÓN

GADUN es una herramienta didáctica desarrollada bajo la supervisión de la UDENAR por el grupo de INVGR como un apoyo al aprendizaje en las asignaturas enfocadas en el diseño de compiladores. Para lo que provee de un entorno en el que se puede emular el diseño de la etapa de análisis de un compilador (análisis léxico y análisis sintáctico).

Para este fin se involucra la definición básica de un analizador léxico impulsado por la librería Regex de Python, e incluye procesamiento descendente utilizando gramáticas LL(1), permitiendo la construcción de dichas gramáticas a través la definición de producciones y símbolos. Además como punto clave para el desarrollo de la herramienta GADUN genera en código Python un proyecto resultante del analizador definido por el usuario.

Como se mencionó con anterioridad la herramienta está orientada al apoyo de cursos introductorios al ámbito de los compiladores, siendo un elemento netamente complementario que busca facilitar al estudiante la visualización e interacción con ciertos conceptos básicos. Por lo que se recomienda su uso a estudiantes o docentes que estén iniciando dentro del espacio de los compiladores, para aquellos que busquen un Meta-compilador robusto, GADUN podría no ser la mejor opción debido a su carácter académico.

# ÍNDICE

<b>INTRODUCCIÓN</b>	<b>3</b>
<b>ÍNDICE</b>	<b>4</b>
<b>OBJETIVOS DEL SISTEMA</b>	<b>5</b>
<b>GUIA DE USO</b>	<b>6</b>
Definición de analizador léxico:	6
Definir Token de Conjunto:	7
Definir Token de Expresión Regular:	9
Modificar y Eliminar Tokens:	10
Ejecutar analizador léxico:	12
¿Qué son los Subset?:	15
Definición de analizador sintáctico:	17
Definición de símbolos No terminales	17
Creación de producciones:	19
Tabla de control y explicación de producciones:	22
Crear error personalizado:	24
Ejecutar analizador sintáctico:	26
Generación de código:	28
Gestión de Proyecto:	30
<b>SOLUCIÓN DE PROBLEMAS</b>	<b>31</b>
Error en definición de tokens	31
Error producciones muertas	31
Error producciones inalcanzables	32
¿Qué son errores de ejecución?	32
<b>SOPORTE</b>	<b>34</b>

## OBJETIVOS DEL SISTEMA

Como sistema GADUN pretende:

- Entender el uso de patrones en la definición de tokens como elementos claves para el análisis léxico.
- Comprender cómo el uso adecuado de símbolos y producciones permiten la construcción de gramáticas que cumplen las características de las LL(1), es esencial para la definición de un analizador sintáctico funcional.
- Asimilar la trazabilidad que existe dentro de un compilador en la definición del analizador léxico y sintáctico.
- Generar proyectos en código python que sirvan como base para la creación de compiladores didácticos más complejos.
- Facilitar la asimilación de los conceptos mencionados y relacionados dentro de un espacio académico.

## GUIA DE USO

GADUN permite la definición de analizadores a través de tres etapas:

1. Definición de analizador léxico:
2. Definición de analizador sintáctico:
3. Generación de código:

### Definición de analizador léxico:

Al ejecutar GADUN la primera vista que se le presentará será la que le permitirá definir el analizador léxico.

Id	Descripción
1	l
2	L
3	d
4	.
5	*
6	+

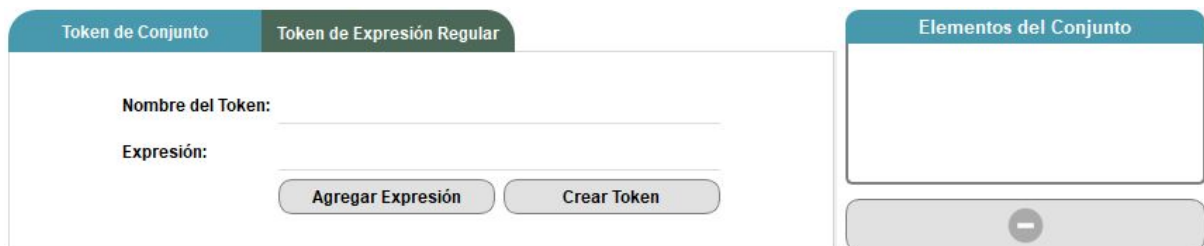
Denote que la sección, '*Analizador Léxico*' está seleccionada en el tab sobre la lista de tokens, indicando que se encuentra en el área de definición para dicho análisis.

Analizador Léxico      Analizador Sintáctico

**Nota:** Puede utilizar este tab para moverse entre las vistas de definición de cada Analizador.

Para la construcción del analizador léxico tenga en cuenta que GADUN utiliza la definición de patrones a los que denominamos tokens, y es en base a estos que se realiza el reconocimiento.

Para definir un token, dirijase a la sección inferior de la vista '*Analizador Léxico*', donde encontrará la sección para definición de tokens. Donde podrá definir patrones para '*Tokens de Conjunto*' y '*Tokens de Expresión Regular*'.



### ***Definir Token de Conjunto:***

Estos tokens son una utilidad de GADUN, diseñados para permitir reconocer expresiones o símbolos específicos que pueden tener alguna relación. Tome por ejemplo un token que denominaremos **operadores** y que reconocerá los símbolos { '+', '-' }.

Primero asegúrese de que la herramienta esté en la sección de '*Token de Conjunto*'.



En la sección '*Nombre de Token*' ubique la cadena que identifica al token, en este caso **operadores**.



Y en la sección '*Expresión*' escriba una por una las expresiones, que reconocerá este token. En este caso empezaremos con '+'.

A continuación presione el botón '*Agregar Expresión*', para agregar la cadena ingresada al token.

**Nota:** El botón situado bajo la lista de '*Elementos del Conjunto*' con el icono (-), puede ser utilizado para remover expresiones del token. Simplemente seleccione en la lista la expresión que desee eliminar y presione el botón.

Si la expresión ingresada es válida, podrá observar como esta fue agregada a la lista de '*Elementos del Conjunto*' situada al lado derecho de la sección de definición de tokens.

Una vez haya terminado de agregar las expresiones que reconocerá el Token, presione el botón de '*Crear Token*'.

Si el proceso ha sido adecuado podrá visualizar en la 'Lista de tokens' encima de la sección de definición de tokens, como se ha agregado el token definido.

Lista de Tokens		
	Nombre	Expresión
1	operadores	+,-



### Definir Token de Expresión Regular:

Como su nombre lo indica estos tokens permiten reconocer secuencias utilizando expresiones regulares. Debido a que GADUN utiliza regex para el análisis léxico las expresiones regulares utilizadas se apegan a las restricciones de esta librería.

Primero asegúrese de que la herramienta esté en la sección de 'Token de Expresión Regular'.



Para la creación de este tipo de tokens al igual que con los de tipo conjunto deberá ingresar un identificador o nombre para el token, y seguido a este la expresión regular que categorice a las expresiones a reconocer.

Tome por ejemplo, un token para reconocer expresiones numéricas al que denominaremos **num**.

Se ingresa en la sección '*Nombre del Token*' el identificador del token, para nuestro caso **num** y en el campo de '*Expresión Regular*' la expresión regular que caracteriza a las secuencias a reconocer, para este caso **[0-9]+** permite reconocer expresiones numéricas de uno o más dígitos. Si la expresión es correcta solo deberá presionar el botón 'Crear Token' para agregar el token a la lista.

Tenga en cuenta la sección del lado derecho de la zona de definición de Tokens, designada como '*Validar Expresión*' para saber si la expresión regular es válida para regex.

Solo podrá agregar el token de expresión regular si la ventana '*Validar Expresión*' indica que la expresión es correcta, de lo contrario obtendrá un mensaje de error.

Si el proceso ha sido correcto podrá observar el token definido en su '*Lista de Tokens*'.

**Lista de Tokens**

	Nombre	Expresión
1	num	[0-9]+

### ***¿Cuándo usar Tokens de Expresión Regular y cuándo usar Tokens de Conjunto?***

Ya que GADUN permite la creación de Tokens de tipo Conjunto, puede surgir la duda de por qué utilizar estos conjuntos y cómo diferenciar su uso de los de Expresión Regular.

La Forma más sencilla de responder a esta cuestión es delegar esto al grado de especificación que se requiera en el reconocimiento. Esto quiere decir que para reconocer expresiones específicas, usar Tokens de tipo conjunto puede facilitar el reconocimiento de estas expresiones. Mientras que para reconocer rangos amplios de expresiones usar los tokens de expresión regular es obligatorio.

Tome por ejemplo, si desea reconocer el conjunto de operadores +, -, \* y / que es un grupo finito y relativamente pequeño de expresiones, usar Tokens de conjunto puede facilitar la creación del token que reconozca estos símbolos.

Nombre	Expresión
operadores	+, -, *, /

*Token de conjunto - Analizador Léxico*

Terminales
operadores:+
operadores:-
operadores:*
operadores:/

*Terminales- Analizador Sintáctico*

Además la transición de estos símbolos en el análisis sintáctico, permitirá el reconocimiento específico de las expresiones en este grupo. en este caso poder reconocer el caso específico puede ser importante debido a la prioridad de estos en la construcción de operaciones.

Por otro lado si este se creará con un token de expresión regular el funcionamiento sería distinto:

Nombre	Expresión
operadores	(\+ \- \* /)

*Token de conjunto - Analizador Léxico*

Terminales
operadores

*Terminales- Analizador Sintáctico*




Por otro lado, conjuntos relativamente más largos como los caracteres alfabéticos o expresiones en las que necesitamos poder utilizar expresiones con cadenas el uso de expresiones regulares es apropiado.

### **Modificar y Eliminar Tokens:**

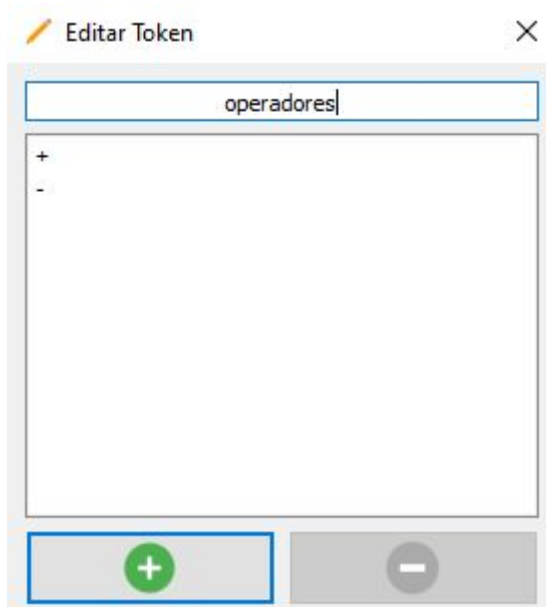
Si desea modificar uno de los tokens que ha definido deberá seleccionar el token a modificar y a continuación presionar el botón con el icono de lápiz en la esquina inferior derecha de la 'Lista de Tokens'.

**Lista de Tokens**

	Nombre	Expresión
1	operadores	+,-
2	num	[0-9]+

 Probar Tokens



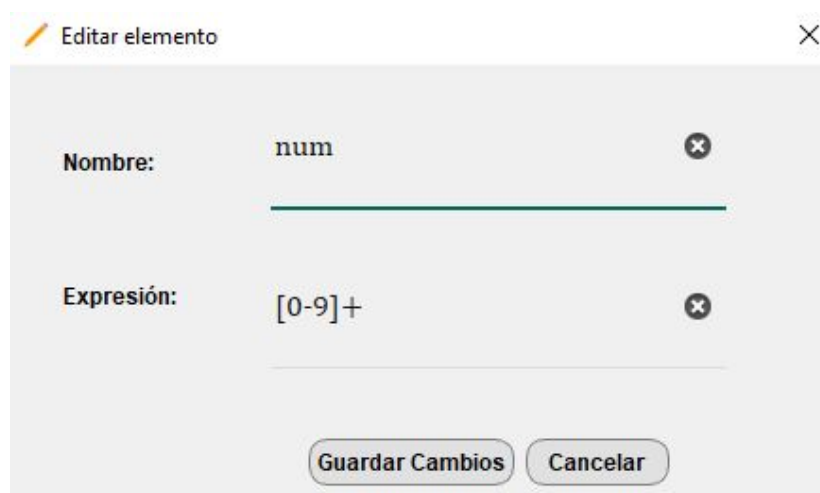
Dependiendo del tipo de Token que se vaya a modificar, se mostrará una ventana de edición diferente. Para este caso **operadores** es un Token de conjunto por lo que la ventana emergente será:

La imagen muestra una ventana de diálogo titulada 'Editar Token' con un icono de lápiz y un botón de cerrar (X). En la parte superior hay un campo de texto que contiene 'operadores'. Debajo de este campo hay una lista desplegable que muestra los caracteres '+' y '-'. En la parte inferior de la ventana hay dos botones: uno con un signo '+' verde y otro con un signo '-' gris.

**Nota:** Los cambios realizados en este tipo de tokens se aplican directamente, por lo que para eliminar una expresión, primero se confirmara si esta está en uso o no. modificar el nombre o identificador del token no requieren confirmación.

De esta manera podrá modificar el conjunto de expresiones utilizando la lista desplegada y los botones (+) y (-) para agregar y eliminar respectivamente expresiones del conjunto. Para modificar el identificador del conjunto simplemente escriba en el campo de texto en que se encuentra el actual identificador(para este caso **operadores**).

Cuando se modifica un token de expresión regular, cómo **num**, la ventana de edición será :




La imagen muestra una ventana de diálogo titulada 'Editar elemento' con un icono de lápiz y un botón de cerrar (X). La ventana tiene dos campos de texto. El primer campo está etiquetado 'Nombre:' y contiene el texto 'num'. El segundo campo está etiquetado 'Expresión:' y contiene el texto '[0-9]+'. Cada campo tiene un botón de eliminar (X) a su derecha. En la parte inferior de la ventana hay dos botones: 'Guardar Cambios' y 'Cancelar'.

Aquí podrá modificar el nombre y la expresión regular del token, para guardar los cambios asegúrese de presionar el botón '*Guardar Cambios*'.

Para eliminar un token, seleccione el token a eliminar y presione el botón con icono (-) en la sección inferior derecha de la 'Lista de Tokens'.

Lista de Tokens

	Nombre	Expresión
1	operadores	+,-
2	num	[0-9] +




 Probar Tokens  

**Ejecutar analizador léxico:**

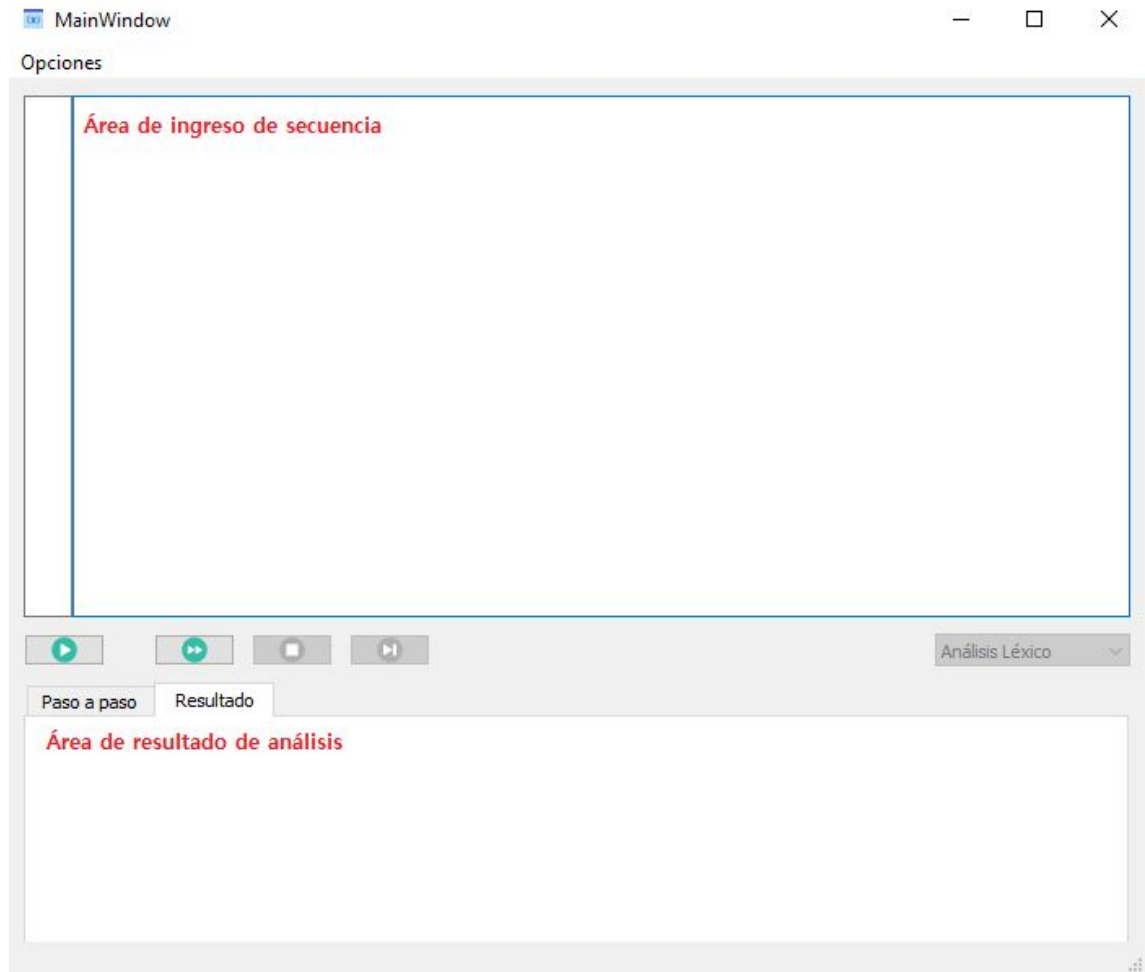
Si desea probar los tokens que ha definido, presione en el botón 'Probar Tokens' en la zona inferior izquierda de la lista de tokens.

Lista de Tokens

	Nombre	Expresión
1	operadores	+,-
2	num	[0-9] +

 Probar Tokens  

La ventana que se despliega es el editor conectado al analizador léxico y sintáctico definidos. Por lo que puede utilizarlo para probar el reconocimiento de estos.



**Nota:** El editor puede ejecutar el análisis lexico y sintactico de manera separada, por lo tanto que se puede indicar el análisis a ejecutar con el desplegable en el costado derecho de la ventana bajo el '*Área de ingreso de secuencia*', en este caso indica '*Análisis Léxico*', opción por defecto.

El editor tiene dos zonas principales el '*Área de ingreso de secuencia*' una zona de texto donde puede ingresar las cadenas a analizar y '*Área de resultado de análisis*' donde se imprimirá el mensaje o resultado del análisis ejecutado.

Para ejecutar el análisis simplemente ingrese una secuencia y presione el botón con el icono play.

Tome por ejemplo los tokens definidos en pasos anteriores:

**operadores:** { +, - }      **num**(expresiones numéricas)

Secuencia en 'Área de ingreso de secuencia' :

1	123 + 97 - 827 --- 9 7 5
---	--------------------------

Secuencia en 'Área de resultado de análisis' :

Paso a paso	Resultado
<b>Aceptado:</b> { num } { operadores:+ } { num } { operadores:- } { num } { operadores:- } { operadores:- } { operadores:- } { num } { num } { num }	

Si el resultado del análisis léxico es aceptado, el mensaje aparecerá en color verde, y el resultado será la cadena traducida o equivalente dada por el analizador léxico basándose en los tokens definidos.

Si se ingresara una secuencia con expresiones no definidas en los tokens, el editor notificará esta situación, tome por ejemplo:

Secuencia en 'Área de ingreso de secuencia' :

1	123 - 97 + palabra
---	--------------------

Secuencia en 'Área de resultado de análisis' :

Paso a paso	Resultado
<b>Error léxico en fila 1 elemento 5:</b> <b>El elemento 'p' no pertenece a los tokens definidos.</b>	

**Nota:** Observe que el mensaje de error indicará el primer error léxico que se encuentre. En este caso el carácter 'p' no coincide con **operadores** o **num**(tokens definidos para el ejemplo), por lo que notifica el error y detiene el análisis

El resultado para este caso indica un error en tono rojo, y la posición en el 'Área de ingreso de secuencia' de este.

1	123 - 97 + palabra
---	--------------------


Adicionalmente en el 'Área de ingreso de secuencia' se resalta la línea con el problema indicado. Para desmarcar el problema en el 'Área de ingreso de secuencia' corrija el error indicado y ejecute nuevamente el análisis.

### ¿Qué son los Subconjuntos?

Los subconjuntos son atajos de expresión regular disponibles por GADUN, se brindan para reducir la cantidad de caracteres utilizados para definir un token de expresión regular.

Subconjunto		
	Id	Descripción
1	l	Cualquier carácter alfabético entre a y z
2	L	Cualquier carácter alfabético entre A y Z
3	d	Cualquier dígito entre 0 y 9
4	.	Cualquier carácter (Excepto, nueva línea)
5	*	Permite reconocer cero o más ocurrencias del ...
6	+	Permite reconocer uno o más ocurrencias del ...

Por defecto GADUN ofrece seis subconjuntos que pueden ser utilizados para crear expresiones regulares.

 Editar elemento Subconjunto ✕

Id:

l

✕

Expresión Python:

[a-z]

✕

Descripción:

Cualquier carácter alfabético entre a y z

Guardar Cambios

Cancelar

Cada subconjunto consta de un **identificador**(carácter que permite utilizar el subconjunto) una **expresión** regular en python(expresión regular a la que equivale para regex) y una breve **descripción**.



Tomando por ejemplo el token anteriormente definido como **num**, para el reconocimiento de expresiones numéricas tenemos:

Expresión regular normal: **[ 0 - 9 ] +**

Expresión regular con subconjuntos: **d +**

Para este caso el intervalo **[0-9]** se reemplaza por la letra **d**(dígito), indicando que se utilizara el subconjunto de dígitos.

Si desea crear un token para reconocer secuencias de caracteres alfabéticos el cambio sería así:

Expresión regular normal: **[ a - z ] +**

Expresión regular con subconjuntos: **l +**

Para este caso el intervalo **[a-z]** se reemplaza por la letra **l**(letra), indicando que se utilizara el subconjunto de letras(minúsculas).

La existencia de estos subsets pueden facilitar la creación de expresiones regulares, pero también implican que para reconocer ciertos caracteres sea necesario utilizar el carácter de escape '**\**', por ejemplo:

Para reconocer la palabra '**dedo**' la expresión regular necesaria sería '**\dedo**'. De esta manera se entiende que la expresión regular debe reconocer el carácter '**d**' y no el subconjunto definido como '**d**'.

**Nota:** Puede crear y modificar los subconjuntos utilizados en su proyecto, pero esto solo se recomienda si se tiene un conocimiento suficiente de expresiones regulares en python regex. De lo contrario podría incurrir en errores de ejecución en el analizador léxico.

### ***Símbolos con significado especial:***

Son símbolos que utilizados dentro de expresiones regulares tienen un significado especial:

**d:** reconoce cualquier carácter numérico entre 0 - 9

**l:** reconoce cualquier carácter alfabético entre a - z

**L:** reconoce cualquier carácter alfabético entre A - Z

**+**: permite la cerradura positiva sobre la expresión anterior

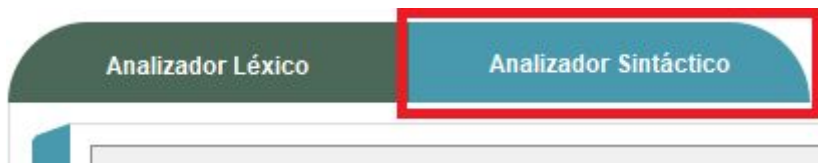
**\***: permite la cerradura de Kleene sobre la expresión anterior

**.**: permite reconocer cualquier carácter

Para reconocer cualquiera de ellos es necesario usar el carácter de escape **\**

## Definición de analizador sintáctico:

Para visualizar el área de definición del analizador sintáctico presione en la opción de 'Analizador Sintáctico' en el tab del área superior de la ventana:



Para definir el funcionamiento del analizador sintáctico tenga en cuenta que GADUN utiliza producciones limitadas para la construcción de gramáticas LL(1) lo que quiere decir que permiten únicamente un elemento no terminal del lado izquierdo.

Antes de empezar a construir las producciones necesitará haber definido unos símbolos terminales y no terminales, como parte de la definición del alfabeto de su gramática.

Para los símbolos terminales bastará con haber definido previamente los tokens del analizador léxico, sin estos la gramática no podrá validarse.

### **Definición de símbolos No terminales**

Por otro lado, para poder definir los símbolos No terminales, deberá dirigirse al área de definición de '*No terminales*' en el costado derecho del área de 'Analizador Sintáctico'.



**Nota:** Los tokens definidos en 'Analizador Léxico' se verán reflejados en símbolos terminales en el 'Analizador Sintáctico'. En la sección inferior a No terminales.

Aquí al presionar el botón 'Agregar' se abrirá una ventana para ingresar el nombre del nuevo No terminal.

The image shows a software interface with two main components. On the left is a dialog box titled 'Añadir No-Terminal' with a close button (X) in the top right corner. Inside the dialog, there is a text input field labeled 'Nombre:' which contains the characters '<>'. This input field is highlighted with a red rectangular border. Below the input field are two buttons: 'Crear' and 'Cancelar'. On the right side of the interface is a panel titled 'No Terminales' which contains an empty list area. At the bottom right of the interface, there are two buttons: 'Agregar' and 'Eliminar'. The 'Agregar' button is highlighted with a red rectangular border.

Aquí deberá ingresar el nombre del no terminal a crear, observe que por defecto se agrega al nombre del No terminal los símbolos '<' y '>' por lo que no es necesario que los escriba.

Una vez presione el botón 'Crear', si el nuevo No terminal no existía previamente se agregara a la lista de No terminales, de lo contrario le notificará la situación.

The image shows the 'No Terminales' panel. At the top is a blue header with the text 'No Terminales'. Below the header is a list containing one entry, '<S>', which is highlighted with a red rectangular border. Below the list, there is a red text message that reads: 'El primer símbolo en la lista siempre sera tomado como el No terminal Inicial'. At the bottom of the panel are two buttons: 'Agregar' and 'Eliminar'.

El primer no terminal en la lista siempre será tomado como el **No Terminal Inicial**, tenga esto en cuenta al definir sus gramáticas.

Si necesita eliminar un No terminal simplemente seleccione el no terminal de la lista y presione el botón 'Eliminar', si este estaba siendo usado en alguna producción desaparecerá de esta.

Con los Terminales y No terminales listos en las listas de símbolos, puede continuar a la construcción de las producciones.

### Creación de producciones:

En el área de lista de producciones se muestran los encabezados 'Lado izquierdo' y 'Lado derecho' como indicadores de a que parte de la producción está relacionado cada campo.

El diagrama muestra una interfaz con dos campos de texto etiquetados como 'Lado Izquierdo' y 'Lado Derecho' con líneas rojas que los conectan a sus respectivos encabezados. A la izquierda del primer campo hay un número '1'. A la derecha del segundo campo hay un botón circular rojo con un signo menos (-).

GADUN utiliza elementos "draggable" lo que significa que para la construcción de las producciones, es necesario arrastrar y soltar los símbolos desde las listas de símbolos hasta sus lugares en los campos respectivos.

El diagrama muestra una interfaz con una lista de símbolos a la izquierda y un campo de producción a la derecha. La lista de símbolos tiene un encabezado 'No Terminales' y contiene el símbolo '<S>'. Debajo del encabezado hay un botón con un signo menos (-). El campo de producción tiene un encabezado 'Lado Izquierdo' y contiene el símbolo '<S>'. Debajo del encabezado hay un botón con un signo plus (+).

Cada una de las líneas representa a una producción de la gramática, así que para agregar más "líneas" o producciones en blanco, solo se necesita presionar el botón con icono (+) en la zona inferior de la lista de producciones.

El diagrama muestra una interfaz con una zona de eliminación y un botón de agregar. La zona de eliminación es un rectángulo con una línea punteada y un icono de papelera. Debajo de ella hay un botón con un signo plus (+) y el texto 'Añadir una producción vacía a la gramática'.

En caso de cometer un error y ubicar un símbolo en un lugar equivocado, solamente debe arrastrar dicho elemento hacia en área con el icono de papelera, para eliminar dicho símbolo.

Para borrar una producción completa deberá presionar el botón con icono (-) que se encuentra del lado derecho de la producción que desea eliminar.

1 <S> -

Como ejemplo ejemplo: retome el analizador léxico definido previamente con los tokens **operadores** y **num**. Se define la siguiente gramática para permitir el reconocimiento de números intercalados por operadores así:

1.  $\langle S \rangle \rightarrow \text{num } \langle OP \rangle \text{ num } \langle S\_L \rangle$
2.  $OP \rightarrow +$
3.  $OP \rightarrow -$
4.  $\langle S\_L \rangle \rightarrow \langle OP \rangle \langle S \rangle$
5.  $\langle S\_L \rangle \rightarrow$

Con lo que se obtiene las listas de elementos terminales y no terminales.

No Terminales	Terminales
$\langle S \rangle$ $\langle OP \rangle$ $\langle S\_L \rangle$	operadores:+ operadores:- num

Denote que el token de expresión regular num ha mantenido su identificador como símbolo terminal, mientras que para el token de conjunto operadores, este se ha dividido en dos símbolos terminales denominados operadores:+ y operadores:- con el objetivo de permitir una mayor especificación en la creación de producciones.

Con estos símbolos ubicados de manera que se asemeje a la gramática indicada anteriormente, se obtiene la siguiente lista de producciones como representación en GADUN:

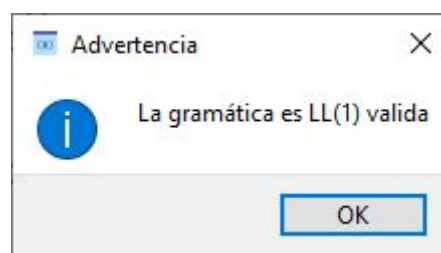
	Lado Izquierdo	Lado Derecho	
1	<S>	num <OP> num <S_L>	-
2	<OP>	operadores: +	-
3	<OP>	operadores: -	-
4	<S_L>	<OP> <S>	-
5	<S_L>		-

Con la lista de producciones definida, se puede continuar con la validación de la gramática. Para ello solo debe presionar el botón Validar producciones debajo del botón para agregar nuevas producciones.

+

▶
Validar Producciones

En caso de obtener una gramática LL(1) válida, aparecerá el mensaje:



De lo contrario aparecerá un mensaje con el error indicado.

Si obtiene en la etapa de validación una gramática LL(1) o incluso una gramática ambigua, podría continuar a la sección de análisis y Tabla de control.

Lado Izquierdo		
Producciones	1	<S>
	2	<OP>
	3	<OP>
	4	<S_L>
	5	<S_L>

#### ***Tabla de control y explicación de producciones:***

En la sección de tabla de control podrá visualizar principalmente la tabla de control resultante de la gramática definida.

	operadores:+	operadores:-	num	↵
<S>	Error	Error	P:0	Error
<OP>	P:1	P:2	Error	Error
<S_L>	P:3	P:3	Error	P:4
num	Error	Error	PA	Error
↵	Error	Error	Error	Acentado

En la tabla de control las intersecciones de fila y columna que contienen una expresión P:N, son donde se aplicaran las producciones con dicho índice. Mientras que las secciones con 'Error' indican estados de error o rechace.

Si se selecciona una de las secciones de aplicación de producción obtendrá:

Explicación de Producción		Selección	Código
Lado Izquierdo	Lado Derecho	Aplicación	
<div>&lt;OP&gt;      operadores:+</div>		<div>pop avanza</div>	
<div>Anulable:                      No</div>			

La primera sección 'Explicación de Producción' indica cómo está conformada la producción a aplicar(lado izquierdo, lado derecho) si esta es anulable y por último cómo se aplica esta en una máquina de pila.

Explicación de Producción	Selección	Código
¿Cómo obtener el conjunto SELECCIÓN?		
Producción	<OP> → operadores:+	
Conjunto Selección	[operadores:+]	
SELECCION(1) = PRIMERO( operadores:+ ) PRIMERO( operadores:+ ) = ( operadores:+ ) SELECCION(1) = ( operadores:+ )		

La sección de 'Selección' indica cual es el conjunto selección de la producción y como este está conformado por el conjunto primero y en el respectivo caso el conjunto siguiente indicado.

Explicación de Producción	Selección	Código
Para empezar se agrega en la zona de análisis sintáctico, un condicional(if o switch) para el "tope de pila" e interno a este otro para la cabecera(puntero) en la cadena analizada, comparados con el lado izquierdo de la producción analizada y su conjunto selección respectivamente. Así:		
<pre> #CONDICIONAL PILA CON LADO IZQ. DE PRODUCCIÓN if (stack[-1] == "&lt;OP&gt;"):     #CONDICIONAL CABECERA CON CONJUNTO SELECCIÓN     if(mainChar[head].token in "[operadores:+]"):         head = self.production1(stack, head)     else:         #SECUENCIA DE ERROR POR DEFECTO           </pre>		



La última sección denominada 'Código' da una breve explicación del código que se agregara al analizador léxico para incluir el funcionamiento de la producción seleccionada.

### **Crear error personalizado:**

Cuando ha definido exitosamente una gramática de características LL(1), podrá editar los mensajes de error o rechace que presenta su gramática, ya que estos por defecto contiene el mensaje genérico de:

"Llegó ELEMENTOX y se esperaba ELEMENTO Y"

Donde el 'elemento X' se refiere al carácter de la cadena analizada que genero problema y 'elemento Y' el carácter o conjunto de caracteres válidos según el conjunto selección definido para dicho caso.

Ejemplo de gramática **operadores, num**:

Paso a paso

Resultado

Error sintáctico en fila 1 elemento 2:

Se encontró num y se esperaba ['operadores:+', 'operadores:-']

Como podrá observar estos mensajes pueden ser muy poco ilustrativos para ciertas situaciones, por lo que GADUN permite al usuario editar el mensaje específico de cada situación de manera que pueda darse una explicación más clara del error.

Para esto en la sección 'Tabla de control' del área de definición de 'Analizador Sintáctico' deberá seleccionar en la tabla de control el error o casilla intersección a la que quiere editar el mensaje.

	operadores:+	operadores:-	num	↵
<S>	Error	Error	P:0	Error
<OP>	P:1	P:2	Error	Error
<S_L>	P:3	P:3	Error	P:4

Tabla de control de gramática **operadores, num**

En la sección inferior a la tabla de control se desplegará una vista para editar el mensaje de la intersección seleccionada.

Mensaje de Error	Código
<b>El caso de error seleccionado por defecto: Se encontró ... y se esperaba ...</b>	
<input type="text" value="Ingrese el mensaje de error para el caso ..."/>	
<b>Cambiar Mensaje</b>	

En el caso de que el error seleccionado mantenga su valor por defecto, este aparecerá con la expresión *'El caso de error seleccionado por defecto...'*, en caso contrario aparecerá el mensaje de error "personalizado".

Para cambiar el mensaje solo deberá ingresar un nuevo mensaje y presionar el botón Cambiar mensaje.

Mensaje de Error	Código
<b>Mensaje de error 1</b>	
<input type="text" value="Mensaje de error 1"/>	
<b>Cambiar Mensaje</b>	

Al hacer esto la etiqueta superior a la zona de ingreso cambiará por el mensaje ingresado.

**Nota:** No es necesario editar todos los mensajes de error, GADUN funciona con mensajes por defecto y personalizados de igual forma.

Con mensajes personalizados o sin ellos si su gramática es validada como LL(1) podrá continuar a la ejecución de su analizador sintáctico.

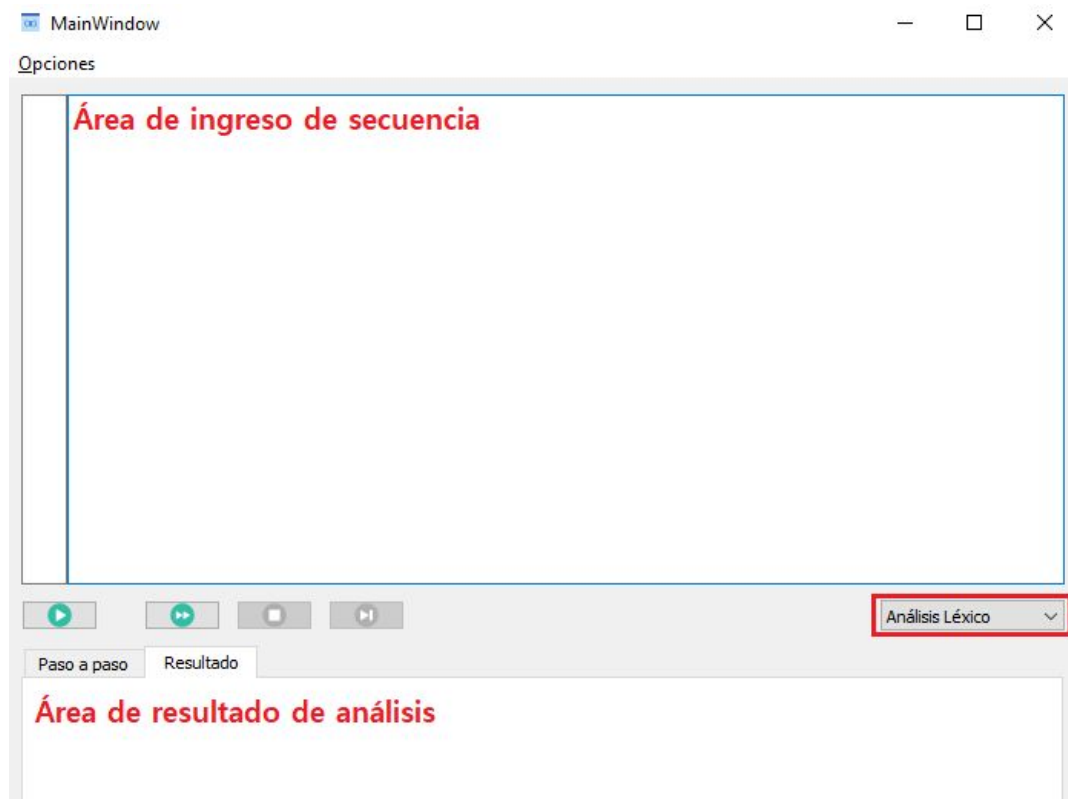
### **Ejecutar analizador sintáctico:**

Para ejecutar su analizador sintáctico ud deberá ingresar a la sección de 'Tabla de Control' en el área de definición de 'Analizador Sintáctico'. Por lo que deberá contar con una gramática válida LL(1). Allí presione el botón 'Abrir Editor' situado en la esquina inferior derecha de la ventana.



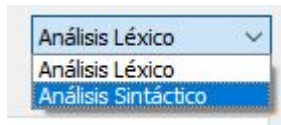
**Nota:** Si el botón aparece 'deshabilitado' significa que ha realizado un cambio en su gramática por lo que debe volver a validarla en el área de producciones presionando el botón 'Validar Producciones'

Una vez presionada aparecerá en su pantalla el editor vinculado al analizador léxico y sintáctico que haya definido.



Como se indicó en definición de analizador léxico el editor tiene dos zonas principales el '*Área de ingreso de secuencia*' una zona de texto donde puede ingresar las cadenas a analizar y '*Área de resultado de análisis*' donde se imprimirá el mensaje o resultado del análisis ejecutado.

Por defecto al iniciar el editor, este está configurado para ejecutar solamente el análisis léxico, por lo que para poder ejecutar el análisis sintáctico deberá especificarlo utilizando el desplegable bajo la esquina inferior derecha de el 'Área de ingreso de secuencia'.



**Nota:** esta función solo está habilitada si el editor se despliega desde la sección de tabla de control.

Una vez seleccionado el analizador sintáctico solamente necesita ingresar una secuencia y presionar el botón con icono play.

Retome el ejemplo de la gramática para **operadores, num**:

Secuencia en 'Área de ingreso de secuencia' :

1	123 + 90 - 76 + 76
---	--------------------

Secuencia en 'Área de resultado de análisis' :

Paso a paso	Resultado
Secuencia aceptada	

Como verá si el resultado del análisis sintáctico es aceptado, el mensaje aparecerá en color verde simplemente indicando que la secuencia es correcta. Por lo contrario, de manera similar a el analizador léxico, si encuentra un error sintáctico, el error aparecerá en un tono rojo y en el 'Área de ingreso de secuencia' se resaltará la línea de error.

1	123 90 - 76 + 76
---	------------------

Paso a paso	Resultado
Error sintáctico en fila 1 elemento 2: Se encontró num y se esperaba [operado	

Ejemplo de error sintáctico, 'Área de ingreso de secuencia' y 'Área de resultado de análisis'

### Ejecución paso a paso:

Como habrá podido observar dentro del editor de GADUN existen botones extra aparte del botón para ejecutar el análisis:



Estos botones permiten la ejecución del análisis paso a paso, este se ejecuta con el botón >> y permite observar elementos extra del proceso de análisis, ya sea sintáctico o léxico.

Paso a paso	Resultado
Elemento en Cabecera <input type="text" value="1237"/>	Cadena Resultado <input type="text" value="num:1237"/>
Token Reconocido <input type="text" value="num"/>	

Ya que estos análisis se realizan por medio de iteraciones el editor muestra el proceso a través de cada una de estas iteraciones.



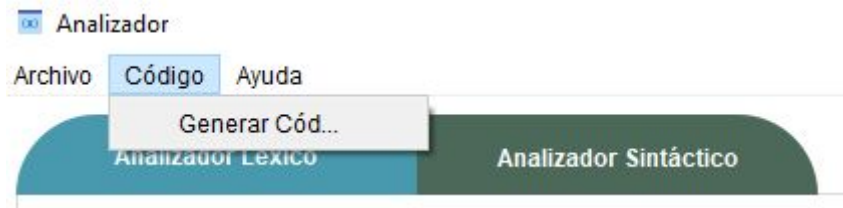
Para ver el siguiente paso de la iteración solo debe presionar >| y para detener el análisis el botón stop.

1237 + 917 - 816 \* 8146

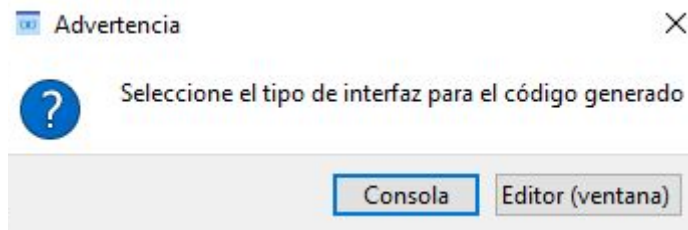
Además el editor también resalta la sección en la secuencia ingresada que está siendo analizada.

## Generación de código:

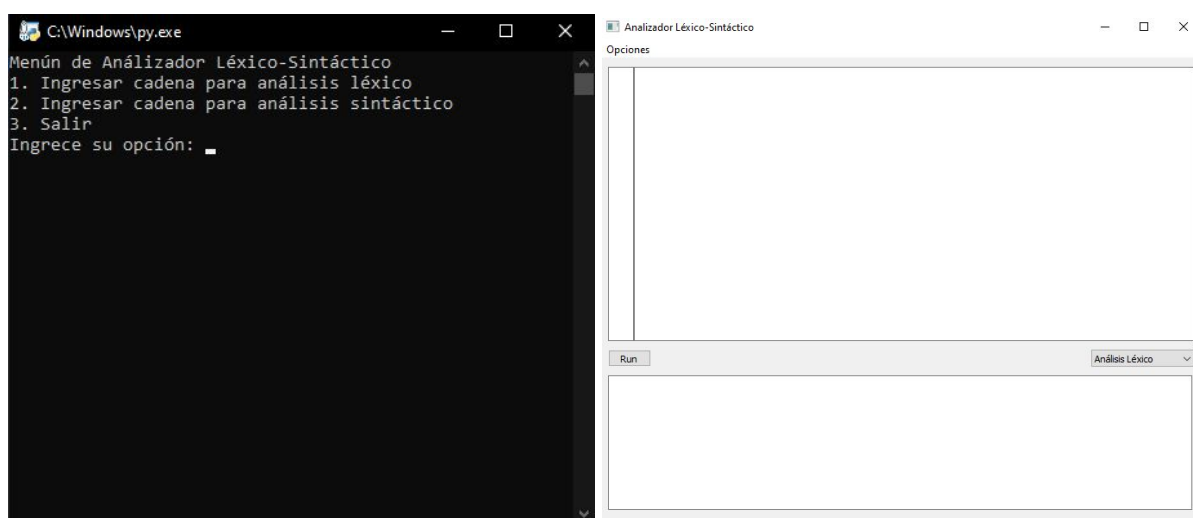
Si ha definido los analizadores léxico y sintáctico o únicamente léxico y ha realizado las pruebas adecuadas su proyecto estará en condición para ser exportado como código python. Para esto en la barra de acciones de la ventana, GADUN le facilita la opción de Generar código de proyecto



Presione la opción indicada y a continuación le aparecerá la ventana solicitando un medio de interfaz para interactuar con su analizador:



Si selecciona 'Consola' su analizador utiliza mensajes de consola para recibir la secuencia a analizar e informar el resultado del análisis. En el caso de seleccionar el editor, su proyecto utilizará una ventana similar a la que ha utilizado como Editor en GADUN.

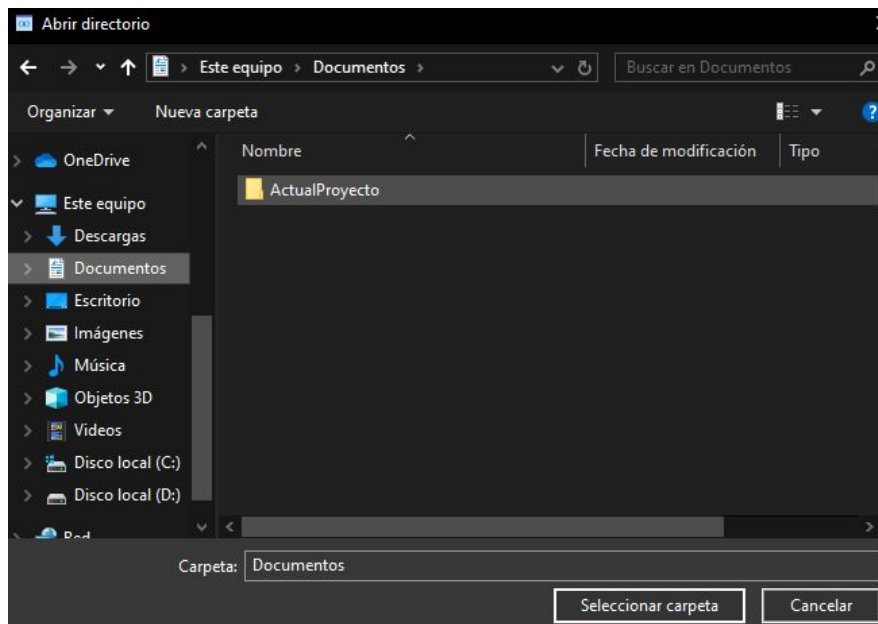


Consola

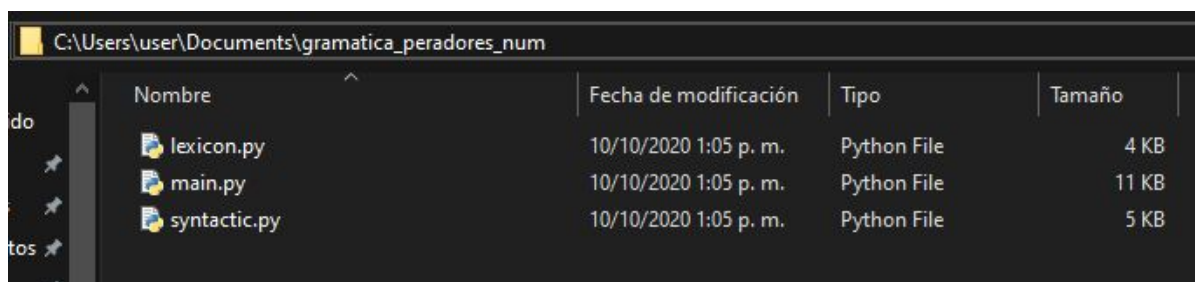
Ventana

La decisión deberá basarse en el medio de interfaz que se desee usar ya que es la única diferencia entre estas dos opciones, tanto el analizador léxico como el sintáctico generado para el proyecto serán iguales en ambos casos.

Es importante denotar que para la generación de código es necesario que su proyecto esté guardado, de manera que GADUN le solicitara un directorio en el cual se creará una carpeta con el código resultando, que tendrá por nombre el nombre de su proyecto.

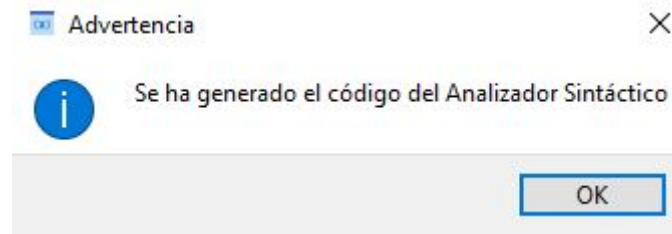


Ejemplo de código generado para gramática **operadores** y **num**:



Observe cómo la carpeta en que contiene su proyecto está conformada por tres archivos; lexicon.py, main.py y syntactic.py. Mientras lexicon y syntactic son los archivos donde puede encontrar la definición de sus analizadores léxico y sintáctico respectivamente. Main es el archivo con el código de la interfaz (consola o ventana) definida para conectar ambos analizadores.

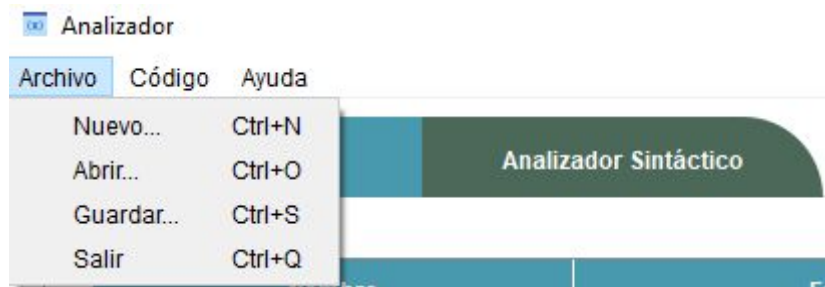
Si el proceso ha sido el adecuado GADUN le notificará que su código ha sido generado.



Dependiendo de si definió el analizador lexico y sintactico o solamente léxico GADUN notificara el estatus del proceso de generación de código. En el caso de ocurrir algún problema lo notificara de la misma manera.

### Gestión de Proyecto:

GADUN gestiona los analizadores a través de proyectos, por lo que permite guardar el estado de los mismos para poder recuperarlos en otro momento.



Para esto solamente utilice el menú de archivo situado en la barra superior derecha de la ventana.



## SOLUCIÓN DE PROBLEMAS

### **Error mi token no funciona como debería**

Los token de expresión regular pueden representar un desafío al definirse para expresiones más específicas, por ello las expresiones regulares resultantes no siempre se comportan como se espera. Si esto sucede con su token de expresión regular, primero verifique si la expresión regular de este en verdad sirve para reconocer el tipo de expresiones deseadas.

Para verificar su expresión regular, ingrese a [PYTHEX](#) y pegue su expresión regular en el primer campo, mientras que en el segundo escriba las expresiones que desea reconocer con esta. Si la función regular no los reconoce, por favor replantee la definición de su expresión regular, tenga en cuenta la documentación en [W3 REGEX](#); si por el contrario esta funciona correctamente utilice la función de reportar bug para dar aviso a soporte de un posible mal funcionamiento

### **Error en definición de tokens**

Si ha definido un token de conjunto o de expresión regular y al ejecutar el analizador léxico en el editor de GADUN, aparece en el 'Área de resultado de análisis' el mensaje:

```
Execution error code: 22  
Error en definición de Tokens
```

Significa que uno de los Tokens definidos no está funcionando correctamente en la librería regex. Se recomienda borrar uno a uno los tokens definidos e ir probando el analizador léxico hasta encontrar el token defectuoso. Para poder corregir su expresión regular.

Para obtener más información sobre expresiones regulares con regex python consulte WWW.

### **Error producciones muertas**




En el proceso de construcción de una gramática pueden cometer equivocaciones que afectan el comportamiento de la misma.



La gramática tiene producciones muertas

Así que para evitar la construcción de analizadores defectuosos la validación de las producciones definidas es un paso fundamental, en el caso de las producciones muertas GADUN indica que uno de los símbolos No terminales genera ciclos infinitos.

En este caso GADUN resaltará las producciones que presentan este problema:

	Lado Izquierdo	Lado Derecho	
1	<A>	letras:a	
2	<B>	letras:b <C>	
3	<C>	letras:c <C>	

Para resolver este problema simplemente inspeccione de arriba hacia abajo las producciones indicadas buscando la producción que genera ciclos infinitos.

#### **Error producciones inalcanzables**

Al igual que el error de producciones muertas esta validación evita errores en el analizador sintáctico.



La gramática tiene producciones inalcanzables

GADUN resaltará las producciones que no pueden ser alcanzadas por medio de derivación desde el no terminal inicial, verifique si la definición de estas producciones es correcta o si el No terminal inicial es el correcto.

	Lado Izquierdo	Lado Derecho	
1	<A>	letras:a	
2	<B>	letras:b <C>	
3	<C>	letras:c	

### ***¿Qué son errores de ejecución?***

En caso de ocurrir uno de estos errores puede haber incurrido en un error lógico de la herramienta, más adelante se describe el punto de error de cada código, aunque lo más apropiado es que se comunique con soporte o envíe un reporte de fallo para reportar un posible bug.

#### **EXECUTION\_ERROR\_4:**

Error en ejecución de análisis de símbolos vivos y muertos.

#### **EXECUTION\_ERROR\_5:**

Error en ejecución de análisis de producciones alcanzables.

#### **EXECUTION\_ERROR\_7:**

Error en ejecución de análisis de producciones ambiguas.

#### **EXECUTION\_ERROR\_8:**

Error en ejecución en primer análisis para generación de conjuntos selección.

#### **EXECUTION\_ERROR\_9:**

Error en ejecución en segundo análisis para generación de conjuntos selección.

#### **EXECUTION\_ERROR\_10:**

Error en ejecución en tercer análisis para generación de conjuntos selección.

#### **EXECUTION\_ERROR\_11:**

Error en ejecución en cuarto análisis para generación de conjuntos selección.

#### **EXECUTION\_ERROR\_12:**

Error en ejecución en quinto análisis para generación de conjuntos selección.

#### **EXECUTION\_ERROR\_13:**

Error en ejecución en sexto análisis para generación de conjuntos selección.

#### **EXECUTION\_ERROR\_14:**

Error en ejecución en séptimo análisis para generación de conjuntos selección.

#### **EXECUTION\_ERROR\_15:**

Error en ejecución en octavo análisis para generación de conjuntos selección.

#### **EXECUTION\_ERROR\_16:**

Error en ejecución en noveno análisis para generación de conjuntos selección.

EXECUTION\_ERROR\_17:

Error en ejecución en décimo análisis para generación de conjuntos selección.

EXECUTION\_ERROR\_18:

Error en ejecución de algoritmo warshall para generación de conjuntos selección.

EXECUTION\_ERROR\_19:

Error en ejecución de operación punto para generación de conjuntos selección.

EXECUTION\_ERROR\_20

Error en ejecución en generación de matriz selección.

## **SOPORTE**

Si presenta problemas de ejecución utilice la opción para reportar error en la sección de ayuda de la herramienta.

O ingrese a <https://github.com/JuanECG/Herramienta-de-apoyo-a-compiladores>