



Sección	Catedrático	Tutor Académico
P	Inga. Asunción Mariana Sic Sor	Elder Anibal Pum Rojas

PROYECTO NO. 2

NodeLex v2.0

DESCRIPCIÓN GENERAL

1. OBJETIVO GENERAL

Que el estudiante cree una herramienta capaz de reconocer un lenguaje, por medio de un analizador léxico que cumpla con las reglas establecidas, manejando la lectura y escritura de archivos, además, de poder implementar un analizador sintáctico que permita la ejecución de instrucciones utilizando una gramática libre de contexto.

2. OBJETIVO ESPECÍFICOS

- Implementar por medio de estados un analizador léxico.
- Utilizar funciones de manejo de cadenas de caracteres en lenguaje Javascript.
- Desarrollar un Scanner para el análisis léxico.
- Construir un scanner basándose en un autómata finito determinístico.
- Crear diagramas con la librería Graphviz
- Implementar un analizador sintáctico utilizando una gramática libre de contexto y su respectivo árbol de derivación.
- Desarrollar una solución web que permita al usuario interactuar de mejor manera con el software.

DESCRIPCIÓN

Actualmente, NodeLex está buscando dejar el campo de la enseñanza estudiantil y se quiere convertir en una herramienta profesional en el campo matemático. La empresa reconoce el potencial del desarrollo web, por lo que, desean migrar del menú interactivo en consola a una solución web, que permita a los usuarios interactuar con el proyecto desde su navegador de preferencia. A su vez, desean tener su propio lenguaje exclusivo con el que el proyecto interactuará y desarrollará los problemas que el usuario desee resolver, para esto, se pidió que se abandone los archivos con extensión **JSON** y se migre a archivos de texto con extensión **NLEX**. Por último, han expresado su deseo de trabajar de forma robusta la información, por lo que, se ha tomado la decisión de implementar un analizador sintáctico que permita verificar de forma lógica, las operaciones matemáticas y de esta forma, poder brindar un servicio de calidad a todos los usuarios nuevos y existentes de NodeLex.

FUNCIONALIDAD

Anteriormente, se trabajaba la información cargando archivos JSON preestablecidos en una ruta concreta, además de, tener un menú sencillo en consola.

Para implementar lo nuevo que la empresa ha solicitado, se requiere manejar archivos dinámicamente, por lo que, es totalmente necesario implementar una solución que permita cargar archivos independientemente de la ubicación, esto con la finalidad de ser más amigable con el usuario.

Los archivos propiamente, deben tener obligatoriamente la extensión **NLEX** para poderse reconocer, eso significa, que el analizador léxico implementado previamente deberá de ser modificado para adaptarse a los nuevos requerimientos que la empresa ha solicitado.

Para la solución de desarrollo web, se ha dejado de libre elección, pero la empresa ha especificado que desea que se desarrolle siempre utilizando un Framework compatible con Javascript, como por ejemplo Angular, React o Vue. Han dejado a elección del desarrollador migrar toda la funcionalidad del analizador léxico a archivos Javascript o Typescript.

Para la implementación del analizador sintáctico, es necesario que el desarrollador realice previamente una gramática libre de contexto y su respectivo árbol de derivación, esto porque la empresa lo ha solicitado como un requerimiento fundamental para su equipo, para poder entrenar a sus empleados con el objetivo de usar el software.

DESCRIPCIÓN DEL LENGUAJE

La forma en la que se cargan las operaciones a trabajar es de la siguiente manera:

```
1 Operaciones = [  
2   {"operacion": "resta", "nombre": "operacion1", "valor1": 6.5, "valor2": 3.5},  
3   {"operacion": "multiplicacion", "nombre": "operacion2", "valor1": 2.3, "valor2": [{"operacion": "seno", "valor1": 90}]},  
4   {"operacion": "suma", "nombre": "operacion3", "valor1": 2.3, "valor2": [{"operacion": "multiplicacion", "valor1": 10, "valor2":  
5     [{"operacion": "raiz", "valor1": 10, "valor2": 2}]}]},  
6 ]
```

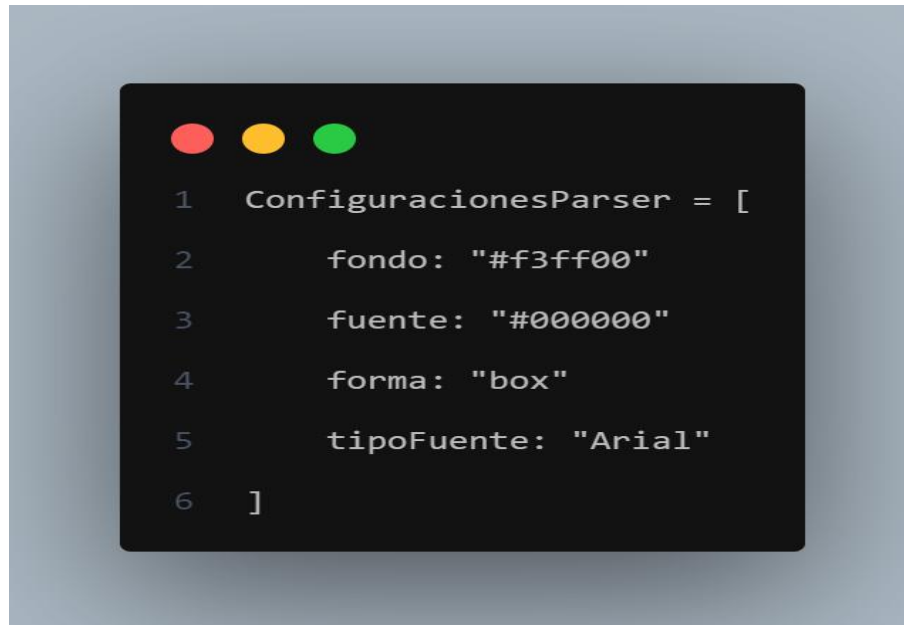
Donde Operaciones es un arreglo de objetos de “operacion”. Cabe resaltar que para esta nueva versión de NodeLex, se permiten N operaciones anidadas, desbloqueando todo el potencial de NodeLex. Además se agrega la propiedad de **Nombre**, que permite agregarle un identificador único a cada operación. Cabe resaltar que solo podrá venir una única propiedad Nombre por operación principal, por lo que, no puede venir una propiedad Nombre en una operación anidada.

Para la parte de las Configuraciones, se trabajan de la siguiente forma:

```
1 ConfiguracionesLex = [  
2   fondo: "#000000",  
3   fuente: "#FFFFFF",  
4   forma: "circle"  
5   tipoFuente: "Times-Roman"  
6 ]
```

En esta nueva versión de NodeLex, se han dividido las configuraciones, una propiedad llamada **ConfiguracionesLex** para el analizador léxico y otra propiedad llamada **ConfiguracionesParser** para el analizador sintáctico.

Para ambas propiedades, se les ha agregado un nuevo atributo llamado **tipoFuente** que sirve para colocar un tipo de fuente en específico, como por ejemplo, Arial o Times-Roman. Además, en esta nueva versión se ha habilitado la opción de manejar colores hexadecimales para los atributos **fondo** y **fuentes**, que permiten colorear el fondo de los nodos de Graphviz y el color de la letra, respectivamente.

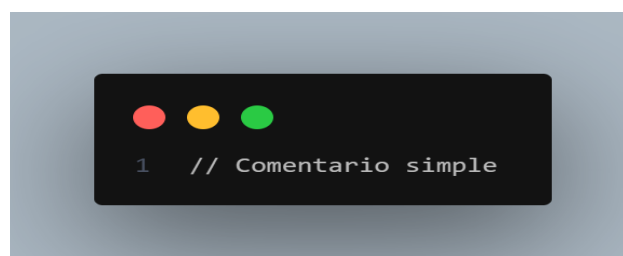
A screenshot of a code editor with a dark background and light blue window control buttons (red, yellow, green) at the top left. The code is written in a light gray font and shows the configuration for 'ConfiguracionesParser' as a JavaScript array with six elements: the array name, a background color, a font color, a shape, a font type, and a closing bracket.

```
1 ConfiguracionesParser = [  
2     fondo: "#f3ff00"  
3     fuente: "#000000"  
4     forma: "box"  
5     tipoFuente: "Arial"  
6 ]
```

Una de las nuevas ventajas de NodeLex v2.0 es la funcionalidad del editor de texto y la propia consola implementada en la página web, por lo que, tenemos funcionalidades de escritura y lectura en el propio editor incorporado. Las nuevas propiedades son:

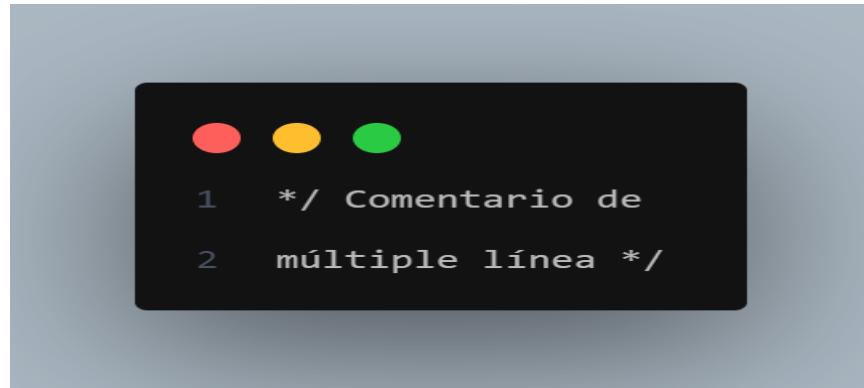
Comentarios:

1. **Comentarios de una línea:** Se representan con doble diagonal o slash y finalizan con un salto de línea.

A screenshot of a code editor with a dark background and light blue window control buttons (red, yellow, green) at the top left. The code is written in a light gray font and shows a single-line comment on the first line of the editor.

```
1 // Comentario simple
```

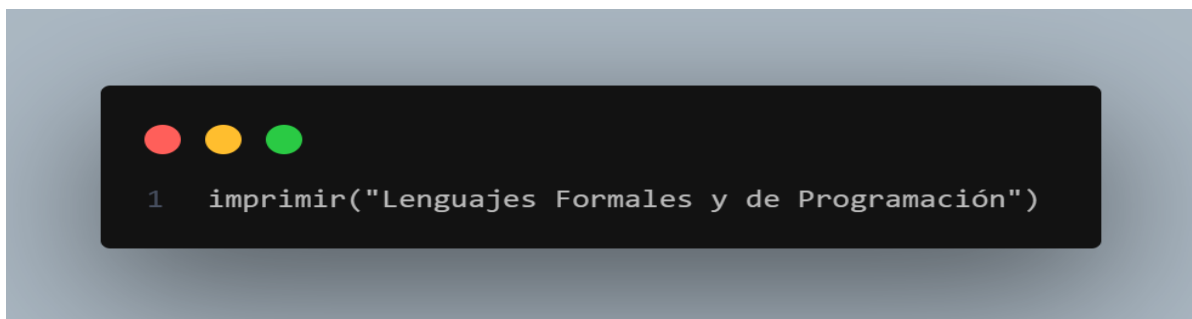
2. **Comentario multilinea:** Se representa de la combinación de asterisco y diagonal y finalizando con la combinación de diagonal y slash.



```
1  */ Comentario de
2  múltiple línea */
```

Instrucciones de Reportería:

1. **Imprimir(cadena):** Imprime en la consola el valor dado por la cadena.



```
1  imprimir("Lenguajes Formales y de Programación")
```

2. **Conteo():** Imprime en la consola el total de operaciones principales que hay en el archivo. Devuelve un número entero.



```
1  conteo()
```

3. **promedio(operacion):** Imprime en consola el promedio de los totales de una operación. Por ejemplo, si existieran 3 operaciones “suma”, lo que hará es sacarle promedio al total de las 3 operaciones “suma”, no importando si son anidadas o no.



4. **max(operación):** Imprime en consola el total máximo de una operación.



5. **min(operación):** Imprime en consola el total mínimo de una operación.



6. **generarReporte(tipoReporte = null, nombreReporte = null)**: Genera los reportes correspondientes. Si no viene nada en el parámetro de tipoReporte, genera todos los reportes de golpe con el siguiente nombre: *número de carnet por defecto, seguido del tipo de reporte que es (201700761_tokens)*. Si en cambio viene la opción de “tokens”, “errores”, “arbol” imprimirá únicamente dicho reporte seleccionado. Además, el usuario puede colocarle el nombre custom si le coloca entre comillas cualquier cadena de texto. El atributo de nombreReporte vendrá únicamente cuando se especifique el tipoReporte.

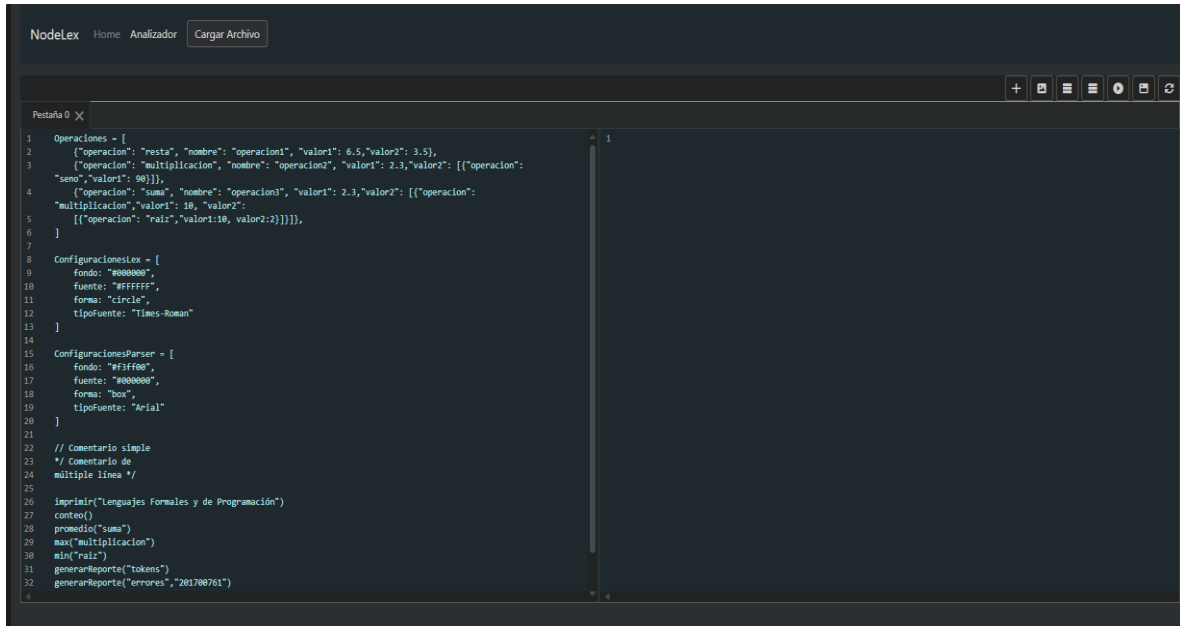


```
1  generarReporte("tokens")
2  generarReporte("errores", "201700761")
3  generarReporte("arbol", "Nueva derivación")
```

INTERFAZ

La interfaz en esta nueva versión migrará de una versión en consola a una versión web, por lo que se solicita utilizar cualquiera de los siguientes Frameworks:

Angular, React, Vue. No se puede utilizar únicamente HTML y CSS, es necesario implementar un framework para la solución correspondiente.



The screenshot shows the NodeLex web interface. At the top, there are navigation links: "NodeLex", "Home", "Analizador", and a button "Cargar Archivo". Below the navigation bar is a toolbar with icons for file operations. The main area is a code editor with a dark theme, showing NLEX code. The code is structured as follows:

```
1 Operaciones = [
2   {"operacion": "resta", "nombre": "operacion1", "valor1": 6.5, "valor2": 3.5},
3   {"operacion": "multiplicacion", "nombre": "operacion2", "valor1": 2.3, "valor2": [{"operacion":
4     "suma", "valor1": 90}],
5   {"operacion": "suma", "nombre": "operacion3", "valor1": 2.3, "valor2": [{"operacion":
6     "multiplicacion", "valor1": 10, "valor2":
7     [{"operacion": "raiz", "valor1": 10, "valor2": 2}]}]
8 ]
9
10 ConfiguracionesLex = [
11   fondo: "#000000",
12   fuente: "#FFFFFF",
13   forma: "circle",
14   tipoFuente: "Times-Roman"
15 ]
16
17 ConfiguracionesParser = [
18   fondo: "#FFFFFF",
19   fuente: "#000000",
20   forma: "box",
21   tipoFuente: "Arial"
22 ]
23
24 // Comentario simple
25 /* Comentario de
26  multiple linea */
27
28 imprimir("Lenguajes Formales y de Programación")
29 conteo()
30 promedio("suma")
31 max("multiplicacion")
32 min("raiz")
33 generarReporte("tokens")
34 generarReporte("errores", "201708761")
35 ]
```

Archivo:

- **Abrir:** Permite abrir un archivo en formato **NLEX**.
- **Guardar:** Permite guardar el archivo que se está editando en nuestra máquina, con formato **NLEX**.
- **Guardar Como:** Permite guardar el archivo que se está editando en nuestra máquina, con el formato que nosotros deseemos.

Analizar: Analizará el texto y mostrará los resultados en la consola incorporada.

OPERACIONES VÁLIDAS

FUNCIÓN	OPERACIÓN
SUMA	Suma de 2 o más números u operaciones anidadas.
RESTA	Resta de 2 o más números u operaciones anidadas.
MULTIPLICACIÓN	Multiplicación de 2 o más números u operaciones anidadas.
DIVISIÓN	División entre números u operaciones anidadas.
POTENCIA	Potencia N de un número u operación anidadas.
RAÍZ	Raíz N de un número u operación anidadas.
INVERSO	Inverso de un número u operación anidadas.
SENO	Función trigonométrica seno de un número u operación anidadas.
COSENO	Función trigonométrica coseno de un número u operación anidadas.
TANGENTE	Función trigonométrica tangente de un número u operación anidadas.
MOD	Residuo entre números u operaciones anidadas.

ESTRUCTURA DEL ARCHIVO DE ENTRADA

```
1 Operaciones = [  
2   {"operacion": "resta", "nombre": "operacion1", "valor1": 6.5, "valor2": 3.5},  
3   {"operacion": "multiplicacion", "nombre": "operacion2", "valor1": 2.3, "valor2": [{"operacion": "seno", "valor1": 90}]},  
4   {"operacion": "suma", "nombre": "operacion3", "valor1": 2.3, "valor2": [{"operacion": "multiplicacion", "valor1": 10, "valor2":  
5     [{"operacion": "raiz", "valor1": 10, "valor2": 2}]}]},  
6 ]  
7  
8 ConfiguracionesLex = [  
9   fondo: "#000000",  
10  fuente: "#FFFFFF",  
11  forma: "circle",  
12  tipoFuente: "Times-Roman"  
13 ]  
14  
15 ConfiguracionesParser = [  
16   fondo: "#f3ff00",  
17   fuente: "#000000",  
18   forma: "box",  
19   tipoFuente: "Arial"  
20 ]  
21  
22 // Comentario simple  
23 */ Comentario de  
24 mltiple línea */  
25  
26 imprimir("Lenguajes Formales y de Programación")  
27 conteo()  
28 promedio("suma")  
29 max("multiplicacion")  
30 min("raiz")  
31 generarReporte("tokens")  
32 generarReporte("errores", "201700761")  
33 generarReporte("arbol", "Nueva derivación")
```

REPORTES

Se necesita realizar los siguientes reportes:

1. Tabla de tokens: Se debe de generar una tabla con todos los tokens analizados indicando el tipo de token, lexema, fila y columna del token leído.
2. Tabla de errores: Se debe de generar una tabla con todos los errores léxicos y sintácticos que se encontraron, indicando el carácter o token leído, fila y columna.
3. Árbol de Derivación: Generado en la lectura del código fuente NLEX utilizando Graphviz.

Para cada uno de ellos, es necesario generar un archivo HTML que describa los valores registrados, en forma de tabla simple, como se muestra a continuación:

Tabla de Tokens

TOKEN	LEXEMA	LÍNEA	COLUMNA
PALABRA	Inicio	1	1
PALABRA	Encabezado	2	5
PALABRA	TituloPagina	3	13
PALABRA	Ejemplo	3	27
PALABRA	titulo	3	35
PALABRA	Cuerpo	5	5
PALABRA	Titulo	6	9
PALABRA	texto	7	13
PALABRA	Este	7	20
PALABRA	es	7	25
PALABRA	un	7	28
PALABRA	titulo	7	31
PALABRA	posicion	8	13
PALABRA	izquierda	8	23
PALABRA	tamaño	9	13
PALABRA	t1	9	21
PALABRA	color	10	13
PALABRA	rojo	10	20
PALABRA	Fondo	12	9
PALABRA	color	13	13
PALABRA	cyan	13	20
PALABRA	Parrafo	15	9
PALABRA	texto	16	13
PALABRA	Este	16	20
PALABRA	es	16	25
PALABRA	un	16	28
PALABRA	parrafo	16	31

ENTREGABLES

En UEDI entregar únicamente el link del repositorio de GitHub que debe incluir:

- Manual técnico en (Markdown de preferencia o PDF)
- Manual de usuario en (Markdown de preferencia o PDF)
- Autómata Finito Determinista (AFD), la expresión regular asociada y el proceso paso a paso del método del Árbol. Esto con el fin de constatar que el estudiante haya aplicado los conocimientos necesarios para desarrollar su proyecto. Puede ir adjuntado al Manual Técnico o en un archivo aparte (Markdown de preferencia o PDF).
- Gramática Libre de Contexto implementada, **en notación BNF**, para verificar que la implementación del analizador sintáctico tenga sentido y corrobore el código implementado. Puede ir adjuntado al Manual Técnico o en un archivo aparte (Markdown de preferencia o PDF).
- Código fuente

CONSIDERACIONES IMPORTANTES

- El proyecto se deberá realizar en forma individual.
- Se debe de crear un repositorio privado en GitHub con el siguiente nombre: LFP_VD_2024_Proyecto2_#Carnet
- El proyecto se implementará en lenguaje Javascript/Typescript utilizando de apoyo Node.js, en caso contrario no será calificado.
- Documentación **OBLIGATORIA**, así como las pruebas del desarrollo del análisis léxico (AFD, expresión regular, método del árbol y gramática libre de contexto). De no ser así, se aplicarán las sanciones respectivas.
- Implementar el uso del archivo package.json, de no ser así, se aplicará una penalización.
- Utilizar uno de los 3 Frameworks sugeridos anteriormente (Angular, React o Vue). Se prohíbe utilizar únicamente HTML y CSS para la parte visual del proyecto.
- Utilizar Programación Orientada a Objetos para el desarrollo del proyecto, de no ser así, se aplicará una penalización.
- No es permitido el uso de ninguna librería para reconocer expresiones regulares ni para la creación del analizador léxico/sintáctico. De ser así, se aplicarán las sanciones respectivas.
- No se aceptan entregas vía correo electrónico u otro medio.
- Se valorará la calidad de la información proporcionada por la aplicación cuando se produzcan errores, así como la presentación y la amigabilidad de la aplicación.
- Sistema Operativo Libre

- Utilizar para generar los diagramas únicamente Graphviz
- Agregar al auxiliar correspondiente de su sección como colaborador a su repositorio Github para poder validar los commits hasta el día de la entrega de su proyecto en UEDI.
- Deberá mantener el repositorio y el permiso disponible hasta finalizado el curso.
- La calificación se realizará en línea y se grabará, esto para que quede constancia de la forma en la que se calificó y como soporte en la toma de decisiones en reclamos por parte del alumno si se presenta el caso.
- La calificación es personal con una duración máxima de 30 minutos, en el horario posteriormente convenido. Durante la calificación el estudiante no podrá modificar el archivo de entrada ni el código fuente de su aplicación.
- El estudiante es responsable del horario que elija para calificarse, en caso de no poder presentarse deberá notificar al auxiliar con suficiente anticipación (1 días antes) para ceder su lugar a otro estudiante, en caso contrario el estudiante solo obtendrá el 80% de su nota obtenida.
- No se dará prórroga para la entrega del proyecto 2.
- **COPIA PARCIAL O TOTAL DE LA PRÁCTICA TENDRÁ UNA NOTA DE 0 PUNTOS, Y SE NOTIFICARÁ A LA ESCUELA DE SISTEMAS PARA QUE SE APLIQUEN LAS SANCIONES CORRESPONDIENTES.**
- En el caso de no cumplir con alguna de las indicaciones antes mencionadas, NO se calificará el proyecto; por lo cual, se tendrá una nota de cero puntos.
- Fecha de entrega: 27 de diciembre de 2024, antes de las 23:59, no se recibirán entregas después de la fecha y hora establecida.