PROYECTO 2



**“Analizador Léxico”**

**LUIS EDUARDO GUILLÉN RUIZ**

UNIVERSIDAD SAN CARLOS DE GUATEMALA

CENTRO UNIVERSITARIO DE OCCIDENTE “CUNOC”

DIVISIÓN DE CIENCIAS DE LA INGENIERÍA

INGENIERÍA EN CIENCIAS Y SISTEMAS

SEGUNDO SEMESTRE

2,019

Ing. OLIVER ERNESTO SIERRA

“ID Y ENSEÑAD A TODOS”

# Contenido del Manual

[Contenido del Manual 2](#_Toc21711603)

[Presentación 3](#_Toc21711604)

[Objetivos del Programa 3](#_Toc21711605)

[Generales 3](#_Toc21711606)

[Específicos 3](#_Toc21711607)

[Instrucciones de Programa 4](#_Toc21711608)

[Requerimientos Técnicos 4](#_Toc21711609)

[Herramientas de Construcción 4](#_Toc21711610)

[Como Acceder al Programa 5](#_Toc21711611)

[Home 5](#_Toc21711612)

[Analizador 6](#_Toc21711613)

[Diagrama de Moore 6](#_Toc21711615)

[Autómata General 7](#_Toc21711616)

[Conjunto de Estados del Autómata: 7](#_Toc21711617)

[Estado Inicial 7](#_Toc21711618)

[Alfabeto ∑ 7](#_Toc21711619)

[Estados de Aceptación F 7](#_Toc21711620)

[Función de Transición (delta) 7](#_Toc21711621)

[Tabla de Transición 8](#_Toc21711622)

[Estado Inicial S0 8](#_Toc21711623)

**“Manual Técnico, Analizador Léxico”**

# Presentación

El presente documento es el manual Técnico del analizador léxico implementando la metodología de creación e identificación de gramáticas por medio de autómatas generados previamente en la cual se le dará acceso al cliente de subir archivos para un análisis en la parte lógica simulando un compilador de lenguaje la cual será la encargada de manejar todos los datos que el cliente desee analizar.

La aplicación está diseñada en NetBeans por medio de la sintaxis Java y se incluye un archivo ejecutable para poder dar la respectiva visualización por medio de cualquier interprete que pueda ejecutar JDK, así mismo se incluye el funcionamiento básico del mismo y una guía para su primera ejecución y posteriormente el uso del mismo.

# Objetivos del Programa

## Generales

* Familiarizar al estudiante con el lenguaje Java.
* Conocer el proceso de análisis léxico realizado por un compilador.
* Aplicar conceptos de creación de Autómatas.
* Aplicar conceptos de aceptación de Gramática.
* Elaborar la lógica para la solución del problema planteado.

## Específicos

* Construcción de algoritmos para los requerimientos de la actividad.
* Ampliar el conocimiento de Programación en Java.
* Desarrollar una aplicación utilizando la metodología de un compilador
* Desarrollar toda la lógica del analizador léxico.
* Desarrollar diagramas de Moore.
* Desarrolló de manual técnico.
* Usar de interfaz gráfica en java, utilizando frames, buttons, entre otros.

# Instrucciones de Programa

1. Instalar JDK.
2. Ejecutar JAR (ejecutable).
3. Ejecutar primeras funciones.
4. Buscar Archivo de Texto a Analizar.
5. Analizar
6. Verificación de Gramática
7. Solicitar Tokens
8. Refrescar
9. Salir

# Requerimientos Técnicos

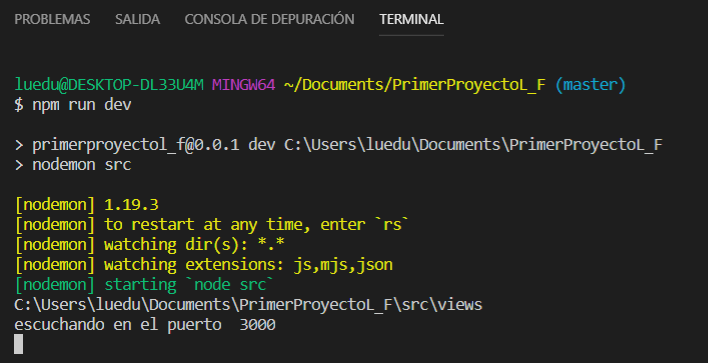
* SO que soporte Java (Ejemp.: Linux, Windows, IOs, Android).
* Java 1.8.0\_201 o compatibles.
* NetBeans IDEA (creación: v. 8.2) o cualquier editor de lenguaje JAVA.
* .Jar ejecutable o proyecto completo.
* Entender lenguaje Java (para lógica del programa).
* Experiencia con aplicaciones Java (funcionamiento).

# Herramientas de Construcción

* NetBeans IDEA
* Package de JAVA
* Clases de JAVA
* Interfaz de JAVA
* PC funcional
* Conocimientos de JavaScript
* Control de Versiones (GitHub)

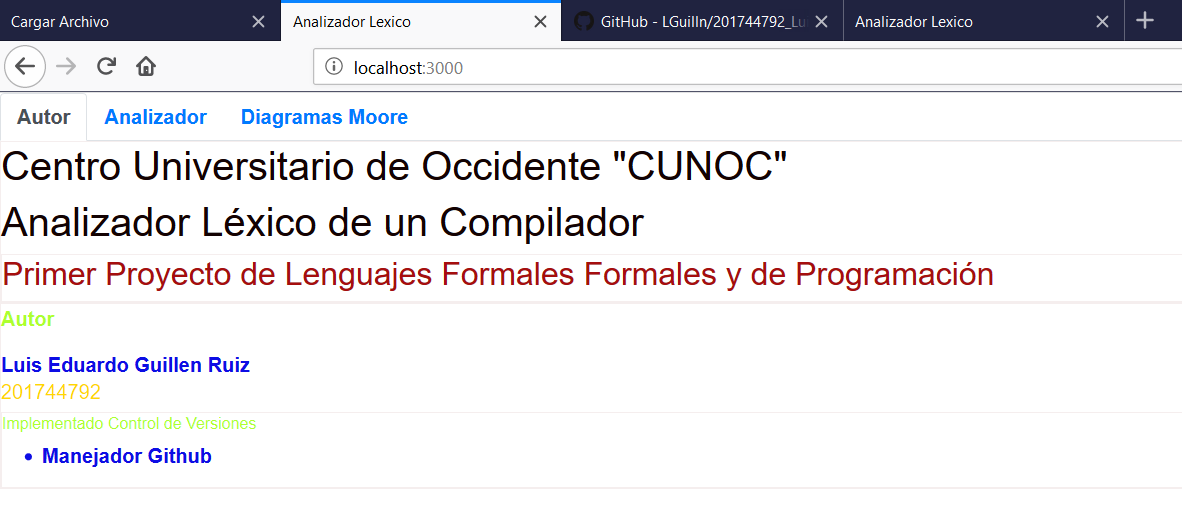
# Como Acceder al Programa

Se debe tener el repositorio Online del Analizador descargar por medio de <https://github.com/LGuilln/201744792_Luis_Guillen_P2>. Luego clonar en repositorio local, instalar requerimientos de java para ejecutar el Jar. Analizar archivos de texto.



Cargar y Ejecutar Proyecto

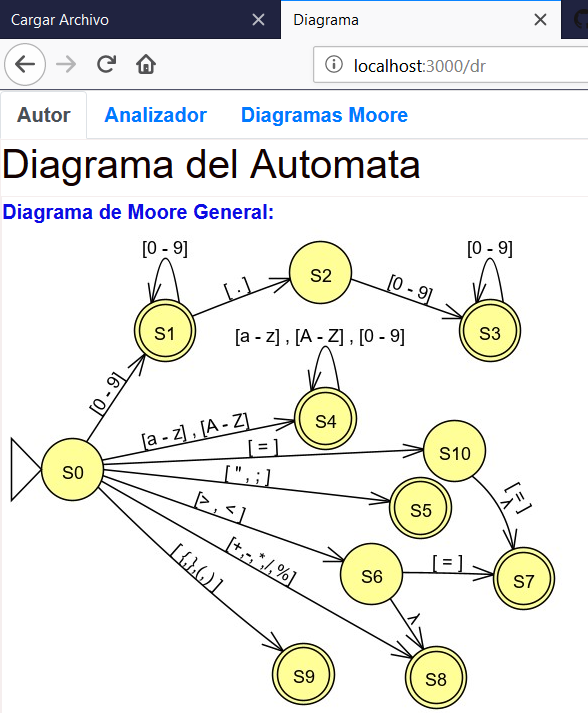
# Home



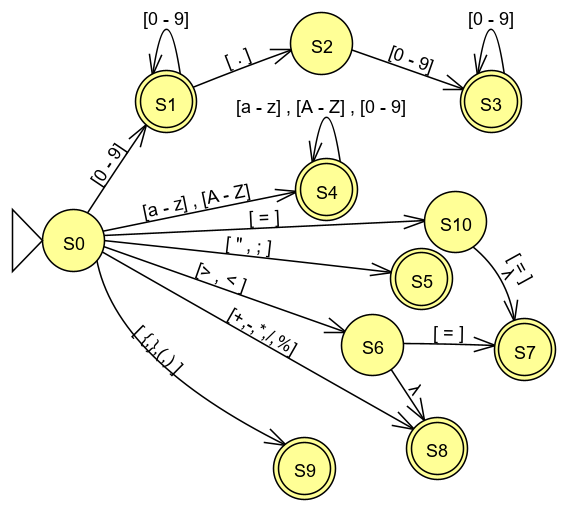
# Analizador

# 

# Diagrama de Moore



# Autómata General

****

## Conjunto de Estados del Autómata:

R//

## Estado Inicial

R// S0

## Alfabeto ∑

R// {[0-9] ⋀ [A-Z] ⋀ [a-z] ⋀ [+, -, /, \*, =, %, <,>] ⋀ [“, ;] ⋀ [{,}, (,)]}

## Estados de Aceptación F

R//

## Función de Transición (delta)

R//

## 

## Tabla de Transición

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **[0-9]** | **[A-Z]** | **[a-z]** | **“, ;** | **< , >** | **+, -, \*, /, %** | **{, }, (, )** | **=** | **.** |  |
| **S0** | S1 | S4 | S4 | S5 | S6 | S8 | S9 | S10 | - | - |
| **S1** | S1 | - | - | - | - | - | - |  | S2 | - |
| **S2** | S3 | - | - | - | - | - | - |  | - | - |
| **S3** | S3 | - | - | - | - | - | - |  | - | - |
| **S4** | S4 | S4 | S4 | - | - | - | - |  | - | - |
| **S6** | - | - | - | - | - | - | - | S7 | - | S8 |
| **S10** | - | - | - | - | - | - | - | S7 | - | S7 |

## Estado Inicial S0

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Cadena** | **[0-9]** | **[A-Z]** | **[a-z]** | **“, ;** | **< , >** | **+, -, \*, /, %** | **{, }, (, )** | **=** | **.** |  |
| **Estado** | S1  S3  S4 | S4 | S4 | S5 | S6 | S8 | S9 | S7  S10 | S2 | S7  S8 |