

## Facultad de Ingeniería

Departamento de Ingeniería de Sistemas

# PRÁCTICA 1 – TEORÍA DE LENGUAJES Y LABORATORIO

## MANUAL DE USUARIO

En el presente documento encontrará la guía con el paso a paso para ejecutar y probar el aplicativo correspondiente a la práctica.

ANTONIO GONZÁLEZ

MATEO RIVERA

Live-Demo: Acá encontrará una versión ya publicada del aplicativo.

http://analizador-lexico-udea.herokuapp.com/

### **Prerrequisitos**

 Node.js/npm: Es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google.

Descarga: <a href="https://nodejs.org/es/download/">https://nodejs.org/es/download/</a>

ó en caso de tenerlo podemos actualizarlo a su versión más reciente ejecutando:

npm install npm@latest -g

 Python 3.9: Es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma.

Descarga: https://www.python.org/downloads/release/python-394/

#### Instalación

1. Clonar el repositorio

git clone https://github.com/antorpo/teoria\_lenguajes\_2021-1.git

2. Ir al directorio de la práctica #1

cd practica\_01

3. Instalar dependencias del back-end

pip install -r requirements.txt

**4.** Migrar la Base de Datos

python manage.py migrate

5. Ir al directorio del front-end

cd frontend

6. Instalar dependencias del front-end

npm install

7. Compilamos la app para producción

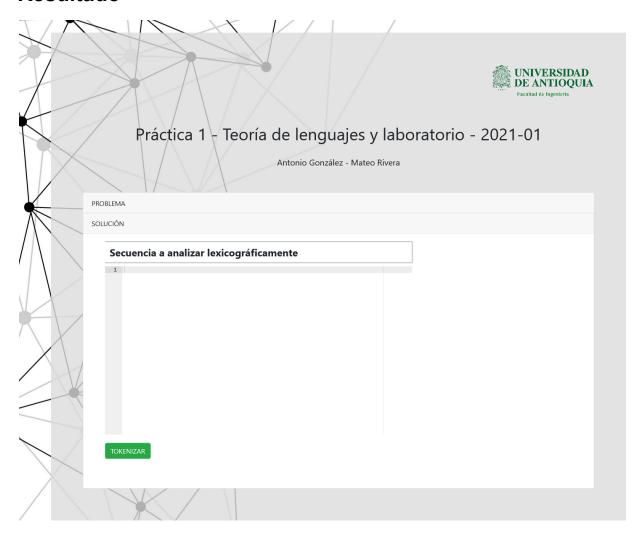
npm run build

 Volvemos al directorio de la práctica #1 y corremos el servidor de desarrollo

cd .. && python manage.py runserver

9. Abrimos el aplicativo en el navegador con ruta localhost:8000/

## Resultado



#### Uso

Al ser tan simple e intuitivo el diseño será muy sencillo utilizar la herramienta, basta con ingresar la secuencia o el fragmento de código a analizar y pulsar en botón de **tokenizar** para visualizar los resultados si el input es válido.

#### Ejemplo-

```
Secuencia a analizar lexicográficamente
  1 if ((condicion1 + condicion2)) {
  2
         funcion1();
  3
    }
  4
  5
  6
  8
  9
 10
 11
 12
 13
 14
 15
 16
 17
 18
 19
 20
 21
 22
 23
 24
 25
TOKENIZAR
```

Al presionar el botón de tokenizar la cadena será procesada en el back-end obteniendo la lista de tokens reconocidos por los autómatas finitos funcionando en conjunto, en este caso lo que llamamos **tokenizador** que evalúa línea a línea y caracter a caracter. Luego de finalizar, los resultados se serializan en formato *.json* y se envían al front-end por medio de un **API** que es consumida para luego ser mostrados en la tabla respectiva que se ve a continuación.

# **Tokens reconocidos**

VALOR	TIPO
if	KEYWORD
(	SEPARATOR
(	SEPARATOR
condicion1	IDENTIFICADOR
+	OPERATOR
condicion2	IDENTIFICADOR
)	SEPARATOR
)	SEPARATOR
{	SEPARATOR
funcion1	IDENTIFICADOR
(	SEPARATOR
)	SEPARATOR
;	SEPARATOR
}	SEPARATOR