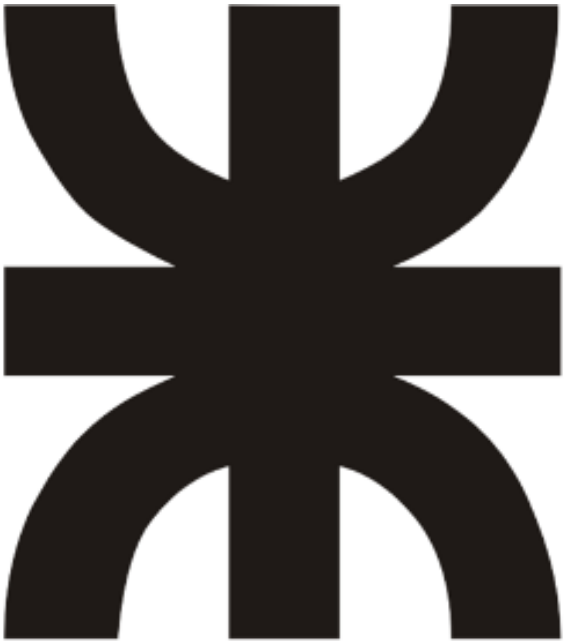
TRABAJO PRÁCTICO INTEGRADOR

Entrega Final- Parser y Lexer - 06/07/2025.

ASIGNATURA: SINTAXIS Y SEMÁNTICA DE LOS LENGUAJES



Cátedra: SINTAXIS Y SEMÁNTICA DE LOS LENGUAJES
Identidad: Los Semánticos.
Cuatrimestre: I
Carrera: ISI.
Fecha: 8/06/25

INTEGRANTES:

- **Chima, Luis Alberto.**
- **Escobar, Ramiro Gabriel.**
- **Fliegler, Franco Emanuel**
- **Gómez, Ricardo Enrique.**
- **Kowtun, Andrea Carolina.**

-2025-

Índice

Introducción

Para esta segunda instancia desarrollamos el Lexer, en donde a través de este documento dejamos asentado las distintas reglas y características fundamentales que debe cumplir, bajo una implementación desarrollada en el lenguaje de programación Python.

Para nosotros es importante explicar que a través de esta entrega parcial detallamos en primera instancia, información del intérprete: cómo los componentes léxicos(tokens) nuestro Lexer analizará del código fuente en JSON ingresado, centrado en las directivas otorgadas por la Cátedra respecto a las características del lenguaje solicitado.

Resumiendo en este trabajo desarrollamos un analizador sintáctico para un conjunto de sentencias en JSON, el cual devolverá las palabras reservadas (tokens) y verificará que la sintaxis sea válida, en caso de que esto no se de así, el programa nos indicará el error.

Objetivo

Utilizando el lenguaje de programación, Python, para el generador de lexer y próximamente el parser se construirán los analizadores léxico (AL) y sintáctico (AS), que permitirán analizar, validar y transformar un subconjunto de etiquetas de la especificación JSON para la descripción de artículos.

Matriz de Habilidades

Tras una primera reunión, se trabajó en nueve aspectos fundamentales para el desarrollo de la actividad. Estos incluyen el conocimiento del idioma inglés, habilidades en programación, producción de video y comunicación. Asimismo, se consideraron otros aspectos clave, como el interés por documentar el trabajo, la capacidad de coordinación, la atención a las indicaciones iniciales de los docentes, la resolución de problemas, la innovación y el pensamiento crítico.

Cada integrante del grupo se identificó con al menos uno de estos aspectos, lo que permitió una distribución equitativa de responsabilidades y un trabajo colaborativo eficiente.

Integrantes/ Aptitudes	Inglés	Programación	Documentación	Producción de Video	Coordinación	Resolución de Problemas	Innovación	Comunicación	Pensamiento crítico
Ramiro Gabriel Escobar	☆	o			o	☆			
Franco Emanuel Fliegler	☆	o	☆	☆	o	o	o	o	o
Luis Alberto Chima		o	o						
Andrea Carolina Kowtun	☆	o	☆		o	o	o	o	o
Ricardo Enrique Gomez	o	o	☆	☆	o		☆	☆	

Identidad

(*“>Los_Semánticos”* Que representa el logo breve)

Se adoptó para la realización del logotipo una temática y paleta de colores al estilo Mátrix, para hacer referencia al apartado de computación/programación en el que nos vamos a encontrar desarrollando el trabajo práctico integrador.

Con el nombre del grupo "Los Semánticos", nombre que hace énfasis redundante en lo a que la materia respecta, representado como si estuviese siendo escrito en algún IDE (Integrated Development Environment) y el ojo simplemente conformando la parte gráfica del logotipo generando así una identidad que abarca las posibilidades:

Solamente Gráfico / Solamente Texto / Combinación entre la parte gráfica y la textual.

Alianza de equipo

Desde la primera reunión del equipo, se estableció como objetivo común adquirir y consolidar el conocimiento de la asignatura a través del desarrollo colaborativo de este trabajo práctico. Para ello, se identificaron las habilidades, disponibilidades horarias y niveles de compromiso de cada integrante, asegurando una distribución eficiente de responsabilidades dentro del grupo.

La metodología de trabajo definida por el equipo consistirá en la realización de reuniones virtuales, con la posibilidad de encuentros presenciales en caso de ser necesario. Estas reuniones se llevarán a cabo los fines de semana en un horario previamente acordado, utilizando plataformas como Google Meet y Discord para facilitar la comunicación y el intercambio de ideas.

Durante cada reunión, se revisan los hitos establecidos en el encuentro anterior y se analizará el avance logrado en el período de trabajo. Cada integrante presentará sus soluciones y expondrá los inconvenientes encontrados, fomentando de esta manera el aprendizaje colectivo y la resolución colaborativa de los problemas.

Como herramienta para la gestión y documentación del trabajo, el equipo ha decidido utilizar Trello, un software de administración de proyectos. A través de esta plataforma, se registrarán las actividades, hitos planificados y el progreso de cada tarea, permitiendo una organización clara y accesible para todos los miembros del grupo.

Analizador Léxico

Analizador léxico

Un analizador Léxico es la primera tarea que realiza un compilador, a su vez es una subrutina del Analizador Sintáctico o Parser.

El lexer es un programa el cual una vez que ingresamos un archivo, lo convierte en una secuencia de caracteres. Luego, los agrupa en palabras con significado propio. Posteriormente, busca los componentes léxicos o palabras que componen el programa fuente, según reglas o patrones.

En otras palabras, el programa Lexer lee carácter por carácter , y de acuerdo a las reglas que definimos agrupa los caracteres y retorna los componentes léxicos que tienen sentido propio dentro del lenguaje. Los caracteres que no logra reconocer los marca como un error.

Veremos a continuación algunas definiciones:

Un TOKEN o COMPONENTE LÉXICO es una unidad léxica indivisible con significado único dentro del lenguaje. Representa la salida del Analizador Léxico. Es la representación usada en el lenguaje.

Un PATRÓN es una expresión regular que permite identificar unívocamente un tipo de token y referenciar al conjunto de todos los tokens que se ajustan a él. Es el conjunto de reglas asociadas a un token.

Un LEXEMA es una cadena de caracteres que encaja con un patrón que describe un componente léxico, es decir, es como una instancia de un patrón. Es una palabra encontrada en el código fuente.

GRAMÁTICA: una gramática consiste de un conjunto de reglas de producción, que pueden ser aplicadas en cualquier orden consistente, con el propósito de derivar un string terminal.

Tareas que realiza un analizador léxico

Las tareas que realizará nuestro Analizador Léxico serán:

- Reconocer los componentes léxicos en la entrada.
- Eliminar los espacios en blanco, los caracteres de tabulación, los saltos de línea y de página y otros caracteres propios del dispositivo de entrada.
- Eliminar los comentarios de entrada.
- Detectar errores léxicos.

Palabras reservadas.

En el caso del lenguaje sobre el cual nos toca realizar el Lexer (DockBook) las palabras reservadas, son consideradas como Tokens.

Procedimiento del AL

Se aplica un método para reconocer los posibles patrones, es decir que, en este caso las expresiones regulares son útiles para describir formalmente los patrones de los tokens.

Se reconocen los tokens, es decir que, los autómatas finitos reconocen los lenguajes por medio de dichas expresiones regulares.

Se realizan acciones preestablecidas al reconocer el token correspondiente.

Errores detectables por el AL

Algunos ejemplos de errores que detecta nuestro analizador léxico son:

- Caracteres ilegales en el programa fuente, es decir, caracteres inválidos, entre otros.

- Errores de ortografía en palabras reservadas. Por lo cual se puede decir que detecta etiquetas mal escritas y también utilización de símbolos no permitidos.

Modo de Obtención del Intérprete

En esta etapa se definieron los Tokens basándonos en la Gramática, con la ayuda de la librería PLY en el lenguaje Python. Después de eso, modificamos el formato y añadimos los tokens específicos de la estructura DocBook para que sean reconocidos por nuestro Lexer. Posteriormente añadimos funciones específicas para una mejor ejecución del programa.

Modo de Ejecución del Intérprete

Los pasos para la ejecución del intérprete son:

1. El usuario localiza el directorio 'bin', donde encontrará un archivo ejecutable de nombre SemanLexer.exe
2. Posteriormente deberá seleccionar algún archivo json de prueba (los proporcionados por nosotros están en el directorio \LosSemánticos-LEXER\doc).
3. El lexer analiza el contenido del JSON, extrayendo los tokens definidos en él.
4. Se muestran los resultados del análisis léxico en la interfaz gráfica, junto con la opción de seleccionar un archivo JSON.
5. Se guardan los tokens extraídos en un archivo .txt externo (LexAnalysis.txt) para futuras referencias.

El intérprete logra:

Funcionar correctamente tanto de forma interactiva como ejecutando las instrucciones desde los archivos de ejemplo.

Indicar como salida si el análisis fue exitoso (archivo correctamente codificado, sin errores), en otro caso indicar los errores existentes (indicando tipo de error, número de línea y cadena que generó el error).

Gramática

TABLA DE TERMINALES Y NO TERMINALES

Token	Valor asociado (ejemplo típico)
FECHA_INICIO	"2024-06-01"
FECHA_FIN	"2024-12-20"
VIDEO	"https://www.youtube.com/watch?v=..."
LLAVE_ABRE	{
LLAVE_CIERRA	}
CORCHETE_ABRE	[
CORCHETE_CIERRA	]
DOS_PUNTOS	:
COMA	,
NUMBER	45, 3000.50
TRUE / FALSE	true, false
EMAIL	"usuario@email.com"
HABILIDADES	"Python, SQL"

SALARIO	4500.75
ACTIVO	true
LINK	"https://sitio.com/perfil.jpg"
NOMBRE	"Juan López"
CARGO	"Developer"
EDAD	30
A_STRING	"Proyecto SSL"
DIRECCION	Nodo interno con "calle", "ciudad"...
EQUIPOS	Lista principal de objetos equipo
TAREAS	Lista de subtareas por proyecto
CONCLUSION	"Documentación completa"
RESUMEN	"Análisis funcional del sistema"
NOMBRE_EQUIPO	"Los Semánticos"
IDENTIDAD_EQUIPO	"SSL2025-A"

REGLAS DE PRODUCCIÓN (Versión más antigua)

Σ → {“EQUIPOS VERSION? FIRMA_DIGITAL?”}

Σ → "{" "equipos" ":" "[" **LISTA_EQUIPOS** "]" VERSION? FIRMA DIGITAL? "}"

VERSION → “versión” “:” A_STRING“,”
VERSION → “versión” “:” null “,”

FIRMA_DIGITAL → “firma_digital” “:” A_STRING
FIRMA_DIGITAL → “firma_digital” “:” null
LISTA_EQUIPOS → EQUIPO|EQUIPO "," LISTA_EQUIPOS

EQUIPO → "{"
"nombre_equipo" ":" A_STRING ","
"identidad_equipo" ":" LINK ","
DIRECCION?
LINK?
"carrera" ":" A_STRING ","
"asignatura" ":" A_STRING ","
"universidad_regional" ":" A_STRING ","
"alianza_equipo" ":" A_STRING ","
"integrantes" ":" "[" LISTA_INTEGRANTES "]" ","
"proyectos" ":" "[" LISTA_PROYECTOS "]"
"}"

LISTA_INTEGRANTES → INTEGRANTE| INTEGRANTE "," LISTA_INTEGRANTES
LISTA_PROYECTOS → PROYECTO| PROYECTO "," LISTA_PROYECTOS

DIRECCION →
"direccion" ":" "{"
"calle" ":" A_STRING ","
"ciudad" ":" A_STRING ","
"país" ":" A_STRING "}" ","

LINK → λ
LINK →null
LINK → Protocolo“://”#string “,”
LINK → Protocolo“://”#string“:”#string “,”
LINK → Protocolo“://”#string“:”#string“/”#string “,”
LINK → Protocolo“://”#string“:”#string“/”#string“/”#string “,”
LINK → PROTOCOLO "://" DOMINIO PUERTO RUTA
LINK → PROTOCOLO "://" DOMINIO RUTA
LINK → PROTOCOLO "://" DOMINIO PUERTO
LINK → PROTOCOLO "://" DOMINIO

PROTOCOLO → "http" | “https” | “ftp” | “ftps”

DOMINIO → SUBDOMINIO A_STRING "." EXTENSION
DOMINIO → A_STRING "." EXTENSION

SUBDOMINIO → “string” "."

A_STRING → “string”

EXTENSION → "com" | "org" | "net" | "edu" | "gov" | "ar" | "es" | "fr"

PUERTO → ":" integer

PUERTO → λ

RUTA → "/" A_STRING RUTA
RUTA → λ

INTEGRANTE → "integrante" ":" "{"
"nombre" ":" A_STRING ","
EDAD?
"cargo" ":" CARGO ","
"foto" ":" LINK ","
"email" ":" EMAIL ","
"habilidades" ":" "["HABILIDADES "]" ","
"salario" ":" float","
"activo" ":" bool
"}"

EDAD → "edad""":"integer | "edad""":" null

CARGO → "Product Analyst" | "Project Manager" | "UX designer" | "Marketing" | "Developer" | "Devops" | "DB admin"

FOTO → LINK
EMAIL → "string""@" "string""." EXTENSION

HABILIDADES → "habilidades" ":" A_STRING

SALARIO → "salario" ":" float
ACTIVO → "activo" ":" bool

PROYECTO → {
"proyecto" : {
"nombre""":" A_STRING,
"estado" ":" ESTADO,
"resumen" ":" "string",
TAREAS,
FECHA_INICIO,
FECHA_FIN,
VIDEO,
"conclusion" : "string"
}
}

ESTADO → "To do" | "In progress" | "Canceled" | "Done" | "On hold" | null

TAREAS → [TAREA] | [TAREA] TAREAS ","

TAREA → "tarea" ":" "{" "nombre""":" A_STRING "estado""":" ESTADO "resumen""":" "string" FECHA_INICIO FECHA_FIN"}"

FECHA_INICIO → "fecha_inicio" ":" "date" ","
FECHA_FIN → "fecha_fin" ":" "date"","
FECHA_FIN → "fecha_fin" ":" null","

VIDEO → LINK

Analizador Sintáctico (Parser)

El parser reconstruye la jerarquía JSON, verificando estructura, claves sensibles a mayúsculas, campos obligatorios y opcionales, secuencia correcta y tipos de datos. Las producciones están conectadas directamente al nodo raíz, evitando reglas inaccesibles. Además, se controla el flujo mediante una producción de errores p_error, mostrando el contexto y token culpable.

Validaciones semánticas

Se implementaron expresiones regulares específicas para:

- Correos electrónicos: deben contener @, dominio y extensión válida de 2 a 4 letras
- URLs: deben empezar con http(s) y contener estructura válida
- Estados permitidos: "To do", "Done", "On hold", etc.
- Cargos válidos de integrante
- Fechas entre 1900 y 2099

Ejecución y modos de uso

El intérprete se ejecuta desde terminal con:

```
python parserLexer.py archivo.json
```

También cuenta con interfaz gráfica usando Tkinter para seleccionar archivos, ver tokens y generar HTML.

Generación de HTML

El archivo HTML se genera dinámicamente:

- Cada equipo dentro de con borde gris
- Datos con <h1>, <h2>, <ul>, <li> según jerarquía
- Links con etiquetas <a> funcionales
- Tareas dentro de tablas <table> con atributos como columnas

Control de errores

Se manejan tres niveles:

Léxicos: caracteres ilegales, mal formato
Sintácticos: claves fuera de lugar, secuencia inválida
Semánticos: correos malformados, estados no válidos, fechas incorrectas

Cada error se muestra con número de línea, tipo y contexto de la cadena afectada.
Restricciones de librerías

limitaciones, consideraciones:

No se usaron librerías como json, html, requests ni frameworks de ayuda. Toda la estructura fue creada desde cero, validando mediante PLY, re para expresiones regulares y Python estándar. La generación de HTML se realizó mediante concatenación de etiquetas manuales, y el análisis de estructura con producciones explícitas.

Problema con las tildes:

Uno de los principales inconvenientes que enfrentamos fue que el analizador sintáctico no lograba avanzar cuando se encontraba con claves con tildes. Esto se debía a que las reglas definidas no contemplaban caracteres con tilde, lo que generaba errores durante el análisis. Si bien adoptamos varias definiciones de tokens para permitir la ejecución del intérprete en los casos previstos, el problema persiste si se presentan nuevas palabras con tildes.

Producciones usadas

Producción	Propósito / Estructura	Ejemplo Representativo
json	Nodo raíz. Acepta equipos + metadatos	{ "equipos": [...], "version": "1.0" }
contenido_json	Variaciones válidas de orden de objetos	Equipos antes o después de version, firma_digital
equipos	Lista de equipos como objetos	"equipos": [{ ... }, { ... }]
lista_equipos	Acumulador de objetos equipo	Dos o más bloques de equipo separados por coma
equipo	Conjunto de campos estructurados del equipo	{ "nombre_equipo": "Los Semánticos", ... }
campos_equipo	Campos de texto, listas o nodos internos (direccion, integrantes)	"carrera": "SSI", "direccion": { ... }
campo_equipo	Campo simple o compuesto. Claves fijas o A_STRING	"link": "https://..." o "identidad_equipo": "ABC"
valor_simple	Primitivos JSON: string, number, boolean, null, date	"Hola mundo", 25, true, null, "2024-06-01"
direccion	Nodo interno que agrupa ciudad, calle, país	"direccion": { "calle": "...", "ciudad": "...", "pais": "..." }
campos_direccion	Campos dentro del objeto direccion	"ciudad": "Resistencia"
campo_direccion	Clave específica: CALLE, CIUDAD, PAIS	"calle": "Av. Libertad"
lista_integrantes	Lista de personas con atributos	"integrantes": [{ ... }, { ... }]
integrante	Objeto con datos como nombre, edad, cargo, etc.	{ "nombre": "Andrea", "edad": 28, ... }

campos_integrante	Campos de tipo valor_simple o campo_email	"email": "usuario@host.com", "salario": 3500.0
campo_integrante	Valida semánticamente email, edad, cargo, links, etc.	"cargo": "Developer" → validado
campo_email	Email como clave y valor asociado en A_STRING	"email": "ejemplo@email.com"
lista_proyectos	Array de proyectos dentro del equipo	[{ ... }, { ... }]
proyecto	Nodo con resumen, estado, fechas y tareas	{ "nombre": "App V2", "estado": "Done", ... }
campos_proyecto	Claves como estado, video, resumen, etc.	"estado": "To do"
campo_proyecto	Incluye validación semántica del estado y links	"video": "https://youtube.com/..."
lista_tareas	Sublista dentro de un proyecto	[{ "nombre": "UX sketch", ... }]
tarea	Objeto con nombre, estado, resumen, fechas	{ "nombre": "Sketch UI", "estado": "Done", ... }
campos_tarea	Campos internos de cada tarea	"resumen": "Diseño de login"
campo_tarea	Validación de estado y link si corresponde	"estado": "Canceled" → validado
version	Campo para metadatos del documento	"version": "2.0"
firma_digital	Campo para autenticación o huella	"firma_digital": "ABCD1234XYZ"

Link Video: **UTN - FRRE - TPI Intérprete JSON a HTML / LOS SEMÁNTICOS**

Conclusión

Este trabajo práctico integrador implicó no solo aplicar conceptos teóricos de análisis léxico y sintáctico, sino también poner en práctica habilidades de diseño, depuración y validación. Desarrollar el intérprete —desde la lectura del JSON hasta la generación de HTML— demandó comprender a fondo la relación entre lexer, parser y semántica.

Definir más de 40 tokens y construir una gramática coherente con la jerarquía del documento llevó múltiples iteraciones. Se resolvieron conflictos sintácticos, se ajustaron reglas y se aplicaron validaciones para correos, fechas, cargos y más.

Un aspecto clave fue trabajar sin librerías externas como `json` o `html`, lo que nos permitió construir una solución desde cero, reforzando la comprensión del funcionamiento de un intérprete real.

También se incorporaron funcionalidades prácticas como ejecución por consola, interfaz gráfica en Tkinter, exportación de tokens y generación visual de HTML. Esto convirtió al proyecto en una herramienta completa, tanto académica como técnica.

En resumen, el trabajo refleja todo lo aprendido en la materia y demuestra lo que se puede lograr combinando teoría, práctica y compromiso con la programación estructurada.

Bibliografía y/o referencias Web

- Documentación oficial de PLY – https://www.dabeaz.com/ply/
- Material académico de UTN FRRe – TPI SSL