ALAN EN TURÍN

Sintaxis y Semántica del Lenguaje

Trabajo Práctico Integrador:
Diseño e implementación
de Lexer y Parser

Intérprete JSON y Traductor a HTML

Ciclo lectivo: 2025

Integrantes:

- Cohen, Alfonsina
- Delvescovo, Benno Matias
- Kundycki, Facundo Kasimierz
- Leconte Revidatti, Santiago
- Vallejos Garcia, Valentino Agustin

Equipo docente:

- Director de Cátedra: Ing. Gabriela P. TOMASELLI
- Docente: Ing. Rodrigo VIGIL
- Docente: Ing. Nicolas G. TORTOSA
- Docente: Ing. Juliana I. TORRE

Índice de Contenido

Introducción

En el marco de la asignatura Sintaxis y Semántica de los Lenguajes, el presente informe representa una instancia clave de la aplicación y consolidación de conocimientos teóricos y prácticos obtenidos a lo largo del curso. El desafío propuesto consiste en el diseño e implementación de un programa capaz de analizar, validar y transformar documentos estructurados en formato JSON, generando como resultado una traducción a lenguaje HTML que represente visualmente los datos procesados. Este proyecto demanda no solo el dominio de técnicas específicas como el análisis léxico y sintáctico, sino también una comprensión integral de los lenguajes formales, estructuras de datos, patrones de diseño y buenas prácticas de desarrollo de software.

Este trabajo tiene como finalidad estimular el pensamiento lógico y la adaptabilidad del grupo, permitiendo así experimentar con la construcción de analizadores léxicos y sintácticos desde cero. La implementación del sistema contempla tanto la validación formal de la estructura del archivo JSON conforme a una gramática definida, como la detección y notificación detallada de errores (léxicos, sintácticos o de ejecución). A su vez, el procesamiento correcto derivará en la generación automática de un documento HTML estructurado, que cumpla con las reglas de presentación especificadas anteriormente y así facilite la interpretación de la información contenida.

Desde la perspectiva pedagógica, el proyecto busca desarrollar competencias vitales en el perfil del futuro ingeniero/a en sistemas, dígase por ejemplo:

- La comprensión profunda de los fundamentos de los lenguajes de programación y de las técnicas de análisis asociadas.

- La capacidad de diseño, planificación y ejecución de proyectos de mediana y alta complejidad, integrando conocimientos de distintas asignaturas previas.

- La promoción de la colaboración efectiva en equipos de trabajo, fomentando la comunicación, la toma de decisiones consensuadas y la responsabilidad compartida.

En este contexto, el presente informe documenta en detalle el proceso de desarrollo del sistema, abarcando desde la conformación del grupo de trabajo y la definición de responsabilidades, hasta la especificación de la gramática, la construcción del lexer y parser, las pruebas realizadas, los mecanismos de control de errores implementados y los criterios seguidos para la generación del HTML final.

Este proyecto no solo refleja el dominio de herramientas y conocimientos técnicos, sino también el compromiso del equipo con la calidad, la organización y la mejora continua en el desarrollo de soluciones informáticas.

Conformación del grupo

Integrantes

- Cohen, Alfonsina
- Delvescovo, Benno Matias
- Kundycki, Facundo Kasimierz
- Leconte Revidatti, Santiago
- Vallejos Garcia, Valentino Agustin

Matriz de habilidades

Para dejar en claro cómo llevaremos a cabo el presente trabajo, confeccionamos una matriz de habilidades que cuenta con aquellas secciones o actividades en la que tengamos facilidad para desenvolvernos, indicando así cierto dominio de las mismas, o que contemos con la iniciativa de querer participar. Utilizaremos los símbolos "★" para indicar el dominio total sobre una competencia; así mismo utilizaremos el símbolo "⚫" para indicar el dominio parcial o iniciativa; aquellas competencias desconocidas por completo sus casilleros permanecerán vacíos.

★ ⚫	Inglés	Programación	Redacción	Edición	Investigación	Cebar mates
Cohen, Alfonsina	★	⚫	★	⚫	★	⚫
Delvescovo, Benno	★	★	★	⚫	★	⚫
Kundycki, Facundo	★	⚫	★	⚫	★	★
Leconte, Santiago	⚫	★	★	⚫	★	★
Vallejos Valentino	⚫	⚫	★	★	★	⚫

Identidad

Acordamos el nombre del grupo **“Alan en Turín”** como un juego de palabras en referencia a Alan Turing, el padre de la ciencia de la computación.
Todo el grupo estuvo de acuerdo con el nombre ya que nos pareció muy original y con mucha personalidad.
Y a su vez para el logo relacionamos el lenguaje binario, Alan Turing y al Mole Antonelliana el edificio más emblemático de Turín, la ciudad de Italia, para crear un logo que tenga todo lo que teníamos en mente.

Alianza de equipo

Como grupo, uno de nuestros principales objetivos es comprometernos a realizar todas las actividades correspondientes en tiempo y forma. Hemos decidido organizarnos fuera del horario de clase, trabajando desde nuestros hogares, ya que consideramos que esta modalidad es la más cómoda y efectiva para desarrollar el proyecto.
Utilizaremos la aplicación Discord como plataforma principal para nuestras reuniones virtuales, ya que nos permite comunicarnos por llamada y compartir pantalla, lo que facilita la exposición de ideas y el trabajo colaborativo. Además, contamos con un grupo de WhatsApp, que será nuestro canal principal para coordinar la disponibilidad de cada integrante y mantener una comunicación fluida.

Gramática

A continuación se detalla la gramática diseñada en base a la estructura del JSON presentada por el escenario. Para realizar esta gramática se requiere que pueda mostrar múltiples niveles de anidamiento, así como también los múltiples tipos de datos que se utilizarán (cadenas de texto, enteros, booleanos, fechas, URLs, entre otros).

El tipo de gramática es libre de contexto o Tipo 2, esto debido a que facilitó el uso de los elementos repetibles u opcionales, así como también las restricciones para los valores predefinidos (como los cargos o los estados de los proyectos). Además, se contemplaron variantes en la representación de listas vacías y estructuras anidadas, todo dentro de una gramática que busca ser lo más clara, completa y cercana posible al formato real de los archivos JSON que se espera analizar.

Derivaciones Iniciales:

Σ → { equipos version firma / equipos firma version / version equipos firma / version firma equipos / firma equipos version / firma version equipos / }

versión → “version:” ““URL”” / { }

firma → “firma:” “String” / { }

Equipos:

equipos → “equipos:“ **[equipo]**

equipo → {“**nombreeq**“: “string”, “**identidadeq**”: “string”, “**direccioneq**”: { } / “string”, “**link**”: { } / ““URL””, “**carrera**”: “string” “**asignatura**”: “string”, “**universidad**”: “string”, “**alianza**“: “string”, “**integrantes**”: *integrantes*, “**proyectos** “: proyectos}

equipo → {“**nombreeq**”: “string”, “**identidadeq**”: “string”, “**direccioneq**”: { } / “string”, “**link**”: { } / ““URL””, “**carrera**”: “string” “**asignatura**”: “string”, “**universidad**”: “string”, “**alianza**“: “string”, “**integrantes**”: integrantes, “**proyectos**“: proyectos}, equipo

Integrantes:

integrantes → “**integrantes**:” [integrante]
integrante → { “**nombre**”: “string”, “**edad**”: Integer / { }, “**cargo**”: “*stringINT*”, “**foto**”: ““URL””, “**email**”: “string”, “**habilidades**”:“string”, “**salario**”: float, “**activo**“: bool }

integrante →{"**nombre**": "string", "**edad**": Integer / { }, "**cargo**": "*stringINT*", "**foto**": ""URL"", "**email**": "string", "**habilidades**": "string", "**salario**": float, "**activo**": bool }, integrante

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

Proyectos:

proyectos → "**proyectos**: " [proyecto]

proyecto → { "**nombre**": "string", "**estado**": "*stringPROY*", "**resumen**": "string", "**tareas**": tareas, "**fecha_inicio**": ""date"", "**fecha_fin**": ""date"", "**video**": ""URL"", "**conclusión** ": "string" }

proyecto → { "**nombre**": "string", "**estado**": bool, "**resumen**": "string", "**tareas**": tareas, "**fecha_inicio**": ""date"", "**fecha_fin**": ""date"", "**video**": ""URL"", "**conclusión**": "string" }, proyecto

stringPROY → To do | In progress | Canceled | Done | On hold

Tareas:

tareas → "tareas: " [tarea]

tarea → {"**nombre**": "string", "**estado**": "string", "**resumen**": "string", "**fecha_inicio**": "date""/ { }, "**fecha_fin**" : ""date""/ { } }

tarea → {"**nombre**" : "string", "**estado**": "string", "**resumen** ": "string", "**fecha_inicio**": ""date""/ { }, "**fecha_fin**" : ""date""/ { } }, tarea

En nuestro caso las producciones con la forma {}, nos referimos a aquellas que pueden tomar valor null o vacío.

Analizador léxico:

Terminado el planteo y creación de la gramática a utilizar en el proyecto, avanzamos en la evaluación de los llamados tokens durante la ejecución del lexer-parsers. En esta segunda etapa del trabajo, debemos seguir con la elaboración de un Analizador Léxico que reconozca tokens, es decir aquellos símbolos terminales que posee la gramática y emitir mensajes de error en el caso de toparse con símbolos inexistentes dentro del alfabeto.

Su manera de operar se basa en leer los caracteres de los ficheros fuente hasta encontrar una entidad con significado léxico, es decir tokens los cuales son una estructura de datos que contiene información acerca del mismo.

```
     tokens = [
          "EQUIPOS", "INTEGRANTES", "DIRECCION", "NOMBRE", "EDAD",
"CARGO", "FOTO", "EMAIL",
          "HABILIDADES", "SALARIO", "ACTIVO", "PROYECTOS", "ESTADO",
"RESUMEN",
          "FECHA_INICIO", "FECHA_FIN", "VIDEO", "CONCLUSION",
"FIRMA_DIGITAL", "NOMBRE_EQUIPO", "IDENTIDAD_EQUIPO",
          "ASIGNATURA", "CARRERA", "UNIVERSIDAD_REGIONAL", "VERSION",
"CALLE", "CIUDAD", "PAIS",
          "ALIANZA_EQUIPO", "LINK", "TAREAS",
          "LLAVE_IZQ", "LLAVE_DER", "CORCHETE_IZQ", "CORCHETE_DER",
"DOS_PUNTOS", "COMA",
          "PUNTO", "ARROBA", "COMILLAS", "BARRA", "GUION",
"GUION_BAJO", "NUMERAL", "ADMIRACION",
          "PORCENTAJE", "AMPERSAND", "APOSTROFE", "PARENTESIS_IZQ",
"PARENTESIS_DER", "ASTERISCO",
          "MAS", "PUNTO_Y_COMA", "MENOR", "IGUAL", "MAYOR",
"INTERROGACION", "BARRA_INV", "ACENTO_CIRCUNFLEJO",
          "BARRA_VERTICAL", "VIRGULILLA", "ACENTO_GRAVE",
          "BOOLEANO", "NULO", "HTTP", "HTTPS", "NUM", "STRING",
"FLOAT",
          "DATE","DOMINIO", "EXTENSION", "RUTA",
          "PROTOCOLO", "PUERTO", "URL", "INTEGER"
     ]
```

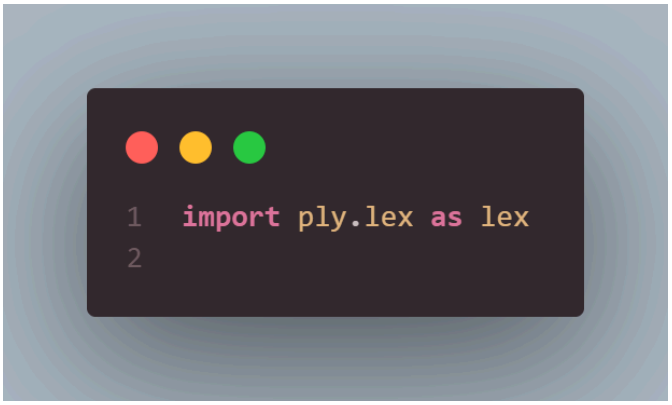
Para esta etapa y durante la continuidad del proyecto se utilizó en la programación el lenguaje python, con la asistencia de la librería como lo son PLY, que permite la creación de analizadores léxicos y sintácticos.

La gramática previa no permaneció inalterable y se utilizaron además funciones que ayuden en el reconocimiento de tokens, los cuales fueron ordenados de manera jerárquica por ser una condición de la librería PLY. Se agregaron dos

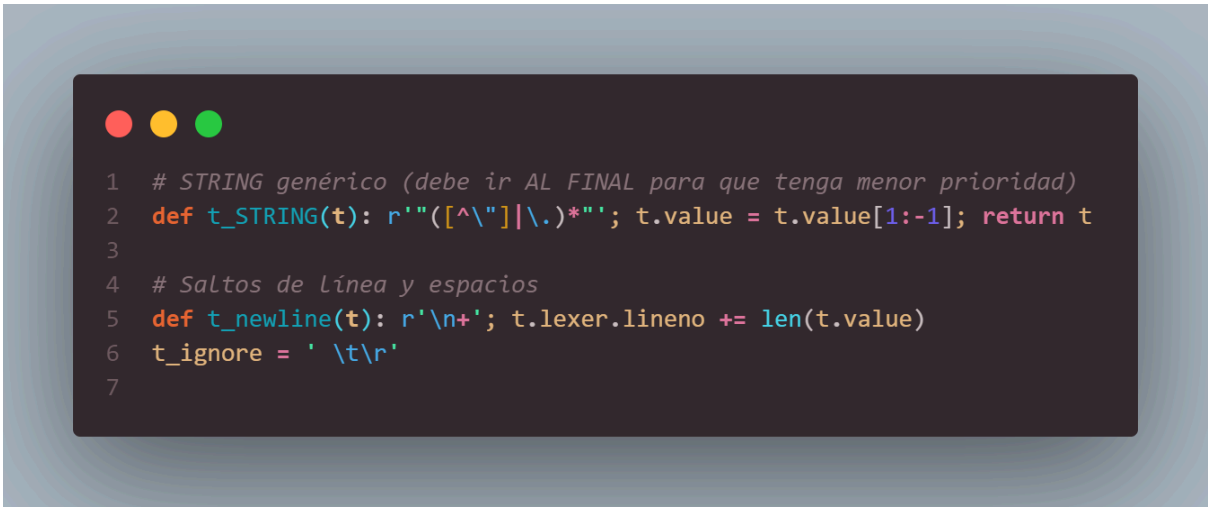
inputs: El primero, es el archivo modelo en JSON otorgado por la cátedra, y luego, realizamos un menú interactivo, en el que se pueden ingresar strings por teclado.

Finalizando la escritura del programa utilizamos PyInstaller para poder así transformar nuestro código en un archivo ejecutable, con sus debidos archivos binarios, y el archivo de texto de prueba. Ofreciendo así, un archivo completamente portable, funcional, listo para cualquier máquina, y fácil de utilizar.

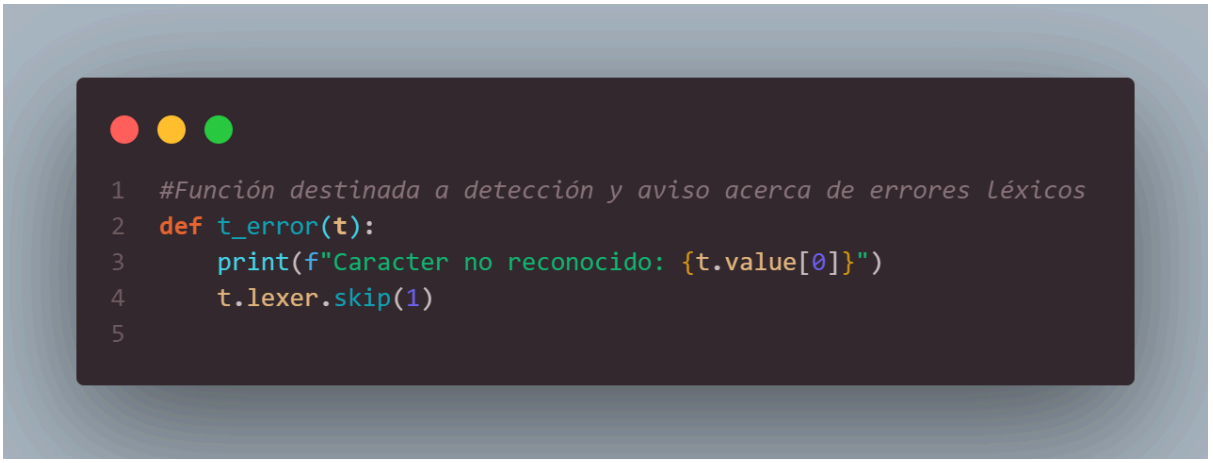
Ejemplos en el código:



```
import ply.lex as lex

```



```
# STRING genérico (debe ir AL FINAL para que tenga menor prioridad)
def t_STRING(t): r'"([^\"]|\.)*"'; t.value = t.value[1:-1]; return t

# Saltos de línea y espacios
def t_newline(t): r'\n+'; t.lexer.lineno += len(t.value)
t_ignore = ' \t\r'

```



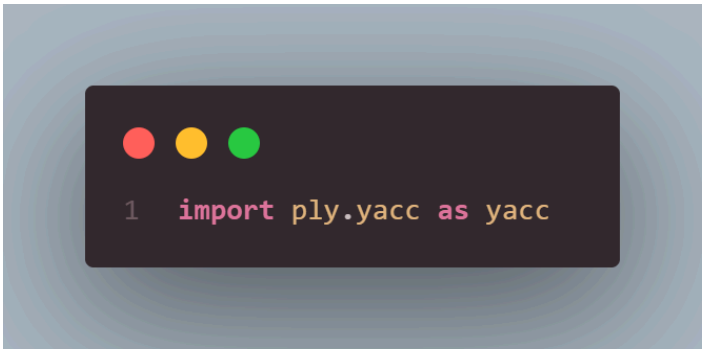
```
#Función destinada a detección y aviso acerca de errores léxicos
def t_error(t):
    print(f"Caracter no reconocido: {t.value[0]}")
    t.lexer.skip(1)

```

```
# Construcción del lexer
lexer = lex.lex()
```

Analizador sintactico:

La tercera y última etapa en este trabajo abarca la confección de un Analizador Sintáctico, el cual pide tokens al Analizador Léxico y los organiza en forma de frases acorde a las reglas que definimos en la gramática. El resultado es un árbol de análisis sintáctico que representa el programa en memoria.

para esto en utilizamos la antes mencionada librería PLY.



```
import ply.yacc as yacc
```

Y se realiza la declaración pertinente para el analizador sintáctico.

```
parser = yacc.yacc()
```

Conclusiones:

El desarrollo de este trabajo práctico integrador nos permitió aplicar de manera concreta los conceptos teóricos que fuimos aprendiendo durante la cursada de Sintaxis y Semántica de los Lenguajes.

Diseñar un analizador léxico y sintáctico desde cero fue un verdadero desafío. No solo requirió pensar en cómo estructurar una gramática válida, sino también en cómo traducirla en código funcional, asegurando que el sistema reconociera correctamente cada parte del archivo JSON y pudiera generar su equivalente en HTML. Trabajar con Python y la librería PLY nos permitió experimentar de forma práctica y útil, una programación más aplicada o avanzada de los proyectos generales que habíamos tenido antes.

También valoramos el trabajo en equipo, la organización y la colaboración entre todos los integrantes. Nos enfrentamos a varios obstáculos en el camino, tanto técnicos como de interpretación, pero gracias al esfuerzo conjunto logramos superarlos y alcanzar un resultado funcional y claro. Además, el uso de herramientas como PyInstaller nos permitió empaquetar el proyecto y convertirlo en una aplicación ejecutable, lo que hizo aún más tangible el trabajo logrado.

En resumen, este TPI no solo nos ayudó a consolidar los conocimientos vistos en clase, sino que también nos permitió ganar experiencia real en el desarrollo de herramientas útiles y estructuradas. Cerramos el proyecto con una sensación positiva, orgullosos de haber transformado los contenidos de la materia en una solución concreta, práctica y funcional.