Universidad **T**ecnológica **N**acional **F**acultad **R**egional **R**esistencia
Departamento de **I**ngeniería en **S**istemas de **I**nformación
Sintaxis y **S**emántica de **L**enguajes

Sintaxis y Semántica del Lenguaje

Trabajo Práctico Integrador:
Diseño e implementación
de Lexer y Parser

Intérprete JSON y Traductor a HTML

Ciclo lectivo: 2024

Equipo docente:

Director de Cátedra

Nombre y Apellido: Ing. Gabriela P. TOMASELLI
Categoría docente: Profesora Asociada
e-mail: gabriela.tomaselli@gmail.com

Docentes

Ing. Gabriela P. TOMASELLI	e-mail: gabriela.tomaselli@gmail.com
Ing. Rodrigo VIGIL	e-mail: rodrigovigil@gmail.com
Ing. Nicolas G. TORTOSA	e-mail: nicotortosa@gmail.com
Ing. Juliana I. TORRE	e-mail: julitorre025@gmail.com

Indice de Contenido

1. Introducción 1

Competencias

El presente trabajo práctico integrador pretende desarrollar las siguientes competencias de la asignatura:

Capacidad para conocer los fundamentos teóricos de los lenguajes de programación y las técnicas de procesamiento léxico, sintáctico asociadas, y saber aplicarlas para la creación, diseño y procesamiento de lenguajes de programación .

Capacidad de diseño, y habilidad instrumental necesaria para llevar a cabo proyectos de cierta envergadura, cuya complejidad exige la utilización de conocimientos adquiridos en diversas asignaturas a lo largo de la carrera.

Aptitud y actitud para trabajar en equipo.

Objetivo

Utilizando algún lenguaje de programación o algún generador de lexer y parser para determinado lenguaje de programación deberán construir los analizadores léxico y sintáctico que permita analizar, validar y transformar un documento JSON que describe y almacena información de empresas y proyectos.
El parser a construir recibe un archivo en formato JSON y deberá indicar si está bien construido (adecuado al formato indicado, sin errores) de otra manera indicar los errores; adicionalmente a medida que analiza el documento deberá transformar el contenido en un documento HTML.

El analizador lexicográfico es un módulo que recibe una secuencia de caracteres que componen el programa a analizar y lo convierte lógicamente en una secuencia de tokens. El analizador sintáctico recibe la secuencia de tokens que le entrega el analizador lexicográfico y verifica si es correcta de acuerdo a una determinada gramática. Adicionalmente puede generar diversas salidas de acuerdo a criterios especificados.

Hay dos grupos de herramientas que se pueden usar para generar los analizadores:
1) Se utilizan expresiones regulares y autómatas finitos para el análisis lexicca que la secuencia pueda ser generada por la gramática del lenguaje.
ográfico y la técnica LALR para el análisis sintáctico. Ejemplos de esto son lex y yacc, que generan código C o C++, o JLex y CUP, que generan código Java. flex y bison son implementaciones libres y gratuitas de lex y yacc.
2) El otro utiliza la técnica LL(k) tanto para el análisis léxico como para el sintáctico, generando parsers descendentes recursivos. Ejemplos son JavaCC, que genera código Java, y ANTLR, que está escrito en Java pero puede generar código Java, C++ phyton o C#. ANTLR se puede conseguir en http://www.antlr.org/.
https://en.wikipedia.org/wiki/Comparison_of_parser_generators

2. Elaboración y entrega

Presentaciones.

Se prevén tres instancias de presentación:

. **1er entrega:** Primera documentación del proyecto; conformación de equipo (matriz, identidad, alianza) y gramática a generar.

Domingo 28 de Abril de 2024

. **2da entrega:** Presentación del lexer o scanner que reconozca los tokens del lenguaje.

Domingo 2 de Junio de 2024

. **3er entrega final:** Presentación de trabajo completo. Lexer y parser, incluye la presentación del trabajo ante la clase con una exposición de no más de 20 minutos.-

Domingo 30 de Junio de 2024

Metodología de entregas parciales y final.

o Un archivo comprimido deberá ser "subido" a la tarea correspondiente dentro del curso en el campus virtual.

o Dicho archivo comprimido deberá contener:

directorio *doc*. Documentación del trabajo (véase el apartado n° 6)
directorio *src*. archivos fuente del proyecto (flex, bison, python,archivos de C (".c", ".h")
directorio *bin*. Archivos binarios listos para ser ejecutados. (lexer y scanner)
directorio *prueba*. Archivos de ejemplo de pseudocódigo con la extensión ".xml"

3. Conformación de equipos

El trabajo se realizará de forma grupal en grupos de hasta cuatro (4) alumnos, donde cada integrante deberá realizar y presentar una parte del trabajo durante la presentación final.
Cada grupo debe ser responsable, autónomo y adaptable.
Cada grupo tendrá la libertad de definir cómo trabajan, pero debe haber un líder que de las pautas de qué se debe hacer, y en qué condiciones se espera que se haga o cuales son los resultados esperados, estará representado por un delegado, encargado de enviar las entregas parciales y el trabajo final al campus virtual, enviará consultas o dudas,etc.

Cada grupo tendrá un docente designado como tutor encargado del seguimiento del trabajo, responderá inquietudes.

Matriz de habilidades

Cada grupo deberá conformar su propia **matriz de habilidades:**

Deberán armar una tabla y escribir en las columnas las principales competencias necesarias para completar el trabajo. En las filas de esa misma tabla los nombres de las personas del equipo.
Luego:
Puntúa dentro de la tabla el grado de habilidad de cada persona respecto a cada competencia:
→ Poner un asterisco * o estrella ★ en donde la persona es experta.
→ Poner un punto ◯ donde la persona no es experta pero podría desenvolverse con dicha competencia (le interesa, es bueno en eso aunque no es su mayor interés, o está dispuesto a aprenderlo).
Deja en blanco competencias que una persona desconozca por completo (o se niegue a participar en ellas).
Ejemplo de Competencia: Inglés, buscar info, programar, idear soluciones, escribir documentación, redes sociales, edición video, cebar mates, etc.-

Identidad

Cada grupo deberá proponer un nombre o logo para identificarse . Algo que represente al grupo y lo haga diferente a los demás.
Un equipo sin identidad nunca será un gran equipo y puede,incluso, que no sea ni equipo.

Alianza de grupo

Un breve enunciado donde el grupo define como trabajará y se organizará, a que se comprometen, cuales son sus metas y valores , los objetivos comunes.
Por ejemplo, si el trabajo requerirá de juntarnos una vez a la semana fuera de la universidad, establecer en qué días y horarios podría ser, o cómo será la organización de esa reunión. Y si necesitaremos comunicarnos fuera de clase, cuáles serán esos canales.

4. Características de lenguaje JSON

Introducción

JSON, abreviatura de "JavaScript Object Notation" (Notación de Objetos de JavaScript), es un formato de intercambio de datos ligero y fácil de leer. Surgió como una alternativa más simple y legible que XML (Extensible Markup Language) para la transmisión de datos entre aplicaciones web. Aunque su nombre hace referencia a JavaScript, JSON es un formato independiente de cualquier lenguaje de programación y es ampliamente utilizado en diversas tecnologías y plataformas.

La estructura de datos en JSON se basa en dos tipos principales: objetos y matrices. Un objeto JSON es una colección no ordenada de pares clave-valor, donde las claves son cadenas de texto y los valores pueden ser cadenas de texto, números, booleanos, otros objetos JSON, matrices o valores nulos. Por otro lado, una matriz JSON es una secuencia ordenada de valores, que pueden ser de cualquier tipo permitido en JSON.

La sintaxis de JSON es muy similar a la notación de objetos de JavaScript, lo que facilita su comprensión y manipulación para desarrolladores familiarizados con ese lenguaje. Consiste en pares de clave-valor encerrados entre llaves {} para objetos y corchetes [] para listas, separados por comas y con los valores representados según su tipo específico.

JSON es utilizado en una variedad de contextos, incluyendo la comunicación de datos entre el cliente y el servidor en aplicaciones web, el almacenamiento de configuraciones y datos estructurados, y como formato de intercambio en servicios de API (Interfaz de Programación de Aplicaciones), debido a su simplicidad, eficiencia y facilidad de lectura y escritura tanto para humanos como para máquinas.

Tipos de Datos:

- String
  - Compuesta por letras, números, signos de puntuación y caracteres especiales.
  - Se deben encerrar entre comillas dobles.
- Integer
  - Números enteros positivos.
  - No llevan comillas.
- Float
  - Número real positivo.
  - A los efectos de este práctico permitir solo 2 posiciones decimales.
  - No llevan comillas.
  - La parte decimal se separa con el punto.
- Bool
  - Booleano, valores posibles **true** o **false**
  - No llevan comillas.
- Date
  - Formato de fecha de tipo “YYYY-MM-DD” (año-mes-día)
  - A los efectos de este práctico limitaremos las fechas de la siguiente manera
    - Año debe ser un número entre 1900 y 2099.
    - Mes debe ser un número entre 1 y 12
    - Día debe ser un número entre 1 y 31
  - No es necesario validar correspondencia entre día y mes (ej. 31 de feb).
  - Se deben encerrar entre comillas dobles.

Estructura del documento

Listas y objetos del documento

- **Opcionales:** Un nombre de elemento sin adornos indica que un elemento debe ocurrir exactamente **una vez**, Cuando un elemento es opcional se representa con ***?***
- **Lista** implica que puede haber más de un elemento de ese objeto.
- I*mportante, todas las keys son sensibles a mayúsculas y deben respetarse como tal.*

Objeto JSON con los atributos: `empresas, version?, firma_digital?`
`empresas` = Lista de objetos empresa.
`empresa` = Objeto con los atributos: `nombre_empresa, fundación, dirección?, ingresos_anuales, pyme, link?, departamentos`.
`dirección` = Objeto con los atributos: `calle, ciudad, país`.
`departamentos` = Lista de objetos departamento.
`departamento` = Objeto con los atributos: `nombre, jefe?,subdepartamentos`
`subdepartamentos` = Lista de objetos subdepartamento.
`subdepartamento` = Objeto con los atributos: `nombre, jefe?, empleados?`
`empleados` = Lista de objetos empleado.
`empleado` = Objeto con los atributos: `nombre, edad?, cargo, salario, activo, fecha_contratacion, proyectos?`
`proyectos` = Lista de objetos proyecto.
`proyecto` = Objeto con los atributos: `nombre, estado?, fecha_inicio, fecha_fin?`

Consideraciones:

- El JSON completo debe estar encerrado entre llaves. {}
- Todos los **objetos** se encierran entre llaves. {}
- Las key de un objeto se encierran entre llaves, mientras que los valores llevan o no llaves dependiendo del tipo de datos (Ver apartado Tipo de datos)
- Las listas se encierran entre corchetes [ ]
- Luego de cada elemento siempre hay una coma **excepto** antes de un cierre de objeto o lista.
- Cuando un elemento es OPCIONAL puede no estar, puede tener valor **null** o puede ser el objeto vacio (**{}** , **[]**)
- Los atributos de los objetos *empresa, empleado* y *proyecto* respetaran el orden dado, mientras que para todos los demás objetos los atributos pueden aparecer en cualquier orden

```
"empresas": [
 {
  "nombre_empresa": string,
  "fundación": integer,
  "dirección": {
   "calle": string,
   "ciudad": string,
   "país": string
  },
  "ingresos_anuales": float,
  "pyme": bool,
```

```
  "link": string,
  "departamentos": [
   {
    "nombre": string,
    "jefe": string,
    "subdepartamentos": [
     {
      "nombre": string,
      "jefe": string,
      "empleados": [
      {
       "nombre": string,
       "edad": integer,
       "cargo": string,
       "salario": float,
       "activo": bool,
       "fecha_contratación": date,
       "proyectos": [
       {
        "nombre": string,
        "estado": string,
        "fecha_inicio": date,
        "fecha_fin": date
       } ]
      } ]
     } ]
   } ]
  }
 ],
 "version": string,
 "firma_digital": string
}
```

Atributos con valores específicos

Si bien JSON solo valida en base al tipo de dato del atributo, a los efectos de este práctico algunos atributos tendrán también restricciones de formato.

link = Tipo de dato string con validación de formato URL. *Ver apartado URL.*
cargo (empleado) = Tipo de dato string, una de las siguientes opciones:

- `Product Analyst | Project Manager | UX designer | Marketing | Developer | Devops | DB admin`

estado (proyecto) = Tipo de dato string, una de las siguientes opciones:

- `To do | In progress | Canceled | Done | On hold`

URL

Nota: Una URL (**Uniform Resource Location**) es la dirección concreta de un recurso en Internet.

La sintaxis completa de una URL es: **protocolo://dominio:puerto/ruta**

- El **protocolo o esquema de red**. Hace referencia al nombre de protocolo de red necesario para poder alcanzar el recurso al que hace referencia la URL. Protocolos habituales son:
  - **http://** (para recursos de la web)
  - **https://** (para recursos de la web contenidos en un servidor seguro)
- **dominio**. Nombre completo en Internet de la máquina (o la red) que posee el recurso en forma de nombre de dominio. No distinguen entre mayúsculas y minúsculas.
- **Puerto.** Opcional. Puerto por el que se debe conectar con el servidor para obtener el recurso. Si no se indica (que es lo habitual) se toma el puerto por defecto. Por ejemplo en *http* el puerto por defecto es el *80*. Si queremos usar uno en particular se indica tras el servidor poniendo dos puntos y el puerto.
- **Ruta.** Opcional. Indica el recorrido dentro de la máquina remota que hay que hacer a través de los directorios para llegar al recurso que queremos. Se pone después del servidor. Ejemplos:
  - */index.html* Accede a la página index.html situada en el directorio raíz.
  - */imagenes/paisajes/foto001.jpg* Accede a la imagen foto001.jpg dentro del directorio paisajes dentro, a su vez, del directorio imagenes.
- Los únicos caracteres permitidos en URL son letras, números, guión medio, guión bajo y punto. además de los caracteres reservados: **# , /, :**

4.1 Traducción a HTML

Traducir el documento, generando un archivo de texto HTML ,transformando algunos elementos en etiquetas HTML. Formateado de la siguiente manera:

- Nombre de archivo: igual al archivo fuente JSON pero con extensión HTM o HTML.
- Empresa: cada empresa debe contenerse en un div ** ** con un borde de 1px gris y un padding de 20px;.
- Nombre de la empresas : encerrados entre tags: **<h1>**
- **Link** deberá traducirse como tag html link **a**, redireccionando al link correspondiente**.**
- Nombre de los departamentos : encerrados entre tags: **<h2>**
- Nombre de los subdepartamentos: encerrados entre tags **<h3>**
- Empleados : deben traducirse como listas html **ul** y los nombres como item **li**
- Proyectos: deben traducirse como tabla debajo del empleado correspondiente, donde cada atributo es una columna.

5. Control de errores

El intérprete deberá detectar, capturar y controlar errores indicando tipo de error , número de línea y cadena que generó el error
• Léxicos:

- Etiquetas faltantes o mal escritas.
- Utilización de símbolos no permitidos.
- Etc.

• Sintácticos:
- Etiquetas que aparecen en lugares no debidos.
- Etiquetas o elementos obligatorios faltantes.
- Elementos con atributos incompatibles.
- Etc.
- Observación : Se valorará la utilización de "reglas de producción de control de errores" que no generen conflictos.

• De ejecución
- archivo de entrada inexistente o con una extensión incorrecta.
- Etc.

5.1. Modos de ejecución del intérprete

El intérprete deberá ejecutarse de dos formas diferentes:

• Modo interactivo - Analiza mientras escribe

- Se escribe e interpreta manualmente desde un terminal de texto.
- Se utilizará el carácter de fin de archivo para terminar la ejecución: *Control + D*
- Útil para reconocer tokens, depuración y pruebas.
- **Obligatorio para segunda presentación (lexer)**

• Ejecución desde un archivo

- Se interpretarán las sentencias de un archivo pasado como argumento desde la línea de comandos , o abierto por opciones gráficas.
- El archivo deberá tener la extensión ".json"
- **Obligatorio para presentación final (parser)**

6. Documentación del trabajo

Se deberá documentar todo el desarrollo de trabajo; recolectando, redactando, describiendo y exponiendo todo lo realizado.

Para ello deberán realizar y presentar:

Informe técnico

Documento principal del trabajo, que resume lo realizado desde el inicio, desde la conformación del grupo hasta la entrega final y exposición. Escrito con sus palabras (No wikipedia, no ChatGPT) , resumiendo y promocionado el trabajo desarrollado, con las siguientes características:

• Portada
- o Título del trabajo desarrollado
- o Nombre y apellidos de las personas que forman el grupo
- o Nombre de la asignatura
- o Nombre de la carrera
- o Primer cuatrimestre
- o Curso académico: 2024
- o Universidad y regional
- o Lugar y fecha

• Índice
- o Las páginas deberán estar numeradas.

• Introducción
- o Breve descripción del trabajo realizado y de las partes del documento.
- o Descripción de cómo se implementó la solución.
- o Información y requerimientos de software para ejecutar y compilar el tp (versiones de compiladores, herramientas, plataforma, etc).

• Conformación del Grupo: Integrantes, matriz, identidad, alianza.

- Gramática.
  - o Descripción de cada regla de la gramática libre de contexto a utilizar.
    - # Símbolos de la gramática
    - # Símbolos terminales (componentes léxicos)
    - # Símbolos no terminales
    - #Reglas de producción de la gramática
      - # Se valorará la inclusión de gráficos explicativos.
- Análisis léxico
  - o Descripción del archivo del lexer para definir y reconocer los componentes léxicos.
- Análisis sintáctico:
  - o Descripción del archivo del parser utilizado para definir la gramática libre de contexto.

- Funciones auxiliares
  - o Se deben indicar y describir las funciones auxiliares que se hayan codificado.
- Modo de obtención del intérprete
  - o Nombre y descripción de cada archivo utilizado
- Modo de ejecución del intérprete
  - o Interactiva
  - o A partir de un archivo
- Conclusiones:
  - o Reflexión sobre el trabajo realizado.
  - o Puntos fuertes y puntos débiles del intérprete desarrollado.
- Bibliografía o referencias web.
- Anexos
  - o Se podrían incluir aquellos anexos que se consideren oportunos para mejora la calidad de la documentación

Video Educativo / Promocional:

Resumen audiovisual de hasta 7 minutos, relatando y presentando con sus palabras el trabajo realizado por el equipo.
Debe resumir y destacar lo documentado en el Informe final.
Debe ser publicado de manera permanente en alguna plataforma de videos (youtube, vimeo, tiktok,etc)

7. Presentación y Criterios de evaluación

Documentación: 40 %

Se tendrá en cuenta lo indicado en el apartado no 6.

- o El código elaborado deberá estar documentado.
- o Se valorará la inclusión de gráficos o figuras.
- o Se valorará la cantidad, originalidad y complejidad de los ejemplos propuestos.
- o También se valorará:
  - # la acentuación,
  - # la corrección ortográfica
  - # y la calidad y claridad de la redacción.
- o Conocimiento que cada integrante del grupo aporte durante la exposición final del trabajo.

Funcionamiento del intérprete (software): 60 %

- o El intérprete deberá:
  - # 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)
  - # Traducir el documento, generando un archivo de texto HTML correcto.

en particular, deberá evaluar y traducir correctamente el ejemplo propuesto por el grupo y dos ejemplos más: uno al azar de los presentados por otros grupos y otro ejemplo elaborado por la cátedra.

o Se valorará
la completitud del lenguaje de código.
La calidad en el diseño del lenguaje y la gramática.
El control de errores.
La ampliación de elementos del lenguaje.

Observación:
 # Además, se valorará la asistencia a clase de prácticas y la resolución de dificultades encontradas durante la elaboración del trabajo.

8. Ejemplos

Básico

```
{
  "empresas": [
    {
      "nombre_empresa": "Mc Donalds",
      "fundación": 2005,
      "dirección": null,
      "ingresos_anuales": 200000.25,
      "pyme": false,
      "departamentos": [
        {
         "nombre": "Ventas",
          "subdepartamentos": [
            {
              "nombre": "Mc Donalds JUC",
              "empleados": [
              {
                "nombre": "Sideshow Mel",
                "cargo": "Product Analyst",
                "salario": 1250.65,
                "activo": true,
                "fecha_contratación": "2023-09-10",
                "proyectos": [
                {
                  "nombre": "Mc Flurry",
```

```
                "fecha_inicio": "2022-01-10",
                "fecha_fin": null
              } ]
            } ]
          } ]
        } ]
    }
  ]
}
```

Ejemplo completo en:

https://frre.cvg.utn.edu.ar/pluginfile.php/155670/mod_resource/content/1/ejemplo_TPI.json