***Asignatura:***

*Sintaxis y Semántica de los Lenguajes*

*Primer cuatrimestre*

***Ciclo Lectivo 2022***

***Universidad Tecnológica Nacional***

*Facultad Regional Resistencia*

*Ingeniería en Sistemas de Información*

**Alumnos:**

**Claro, Michelle Andrea**

**Saucedo, Gonzalo Nicolás**

**Paredes, Samuel Octavio**

**Ramírez, Eduardo Manuel**

**Docentes:**

**Tomaselli, Gabriela**

**Torre, Juliana**

**Tortosa, Nicolás**

**Vigil, Rodrigo**

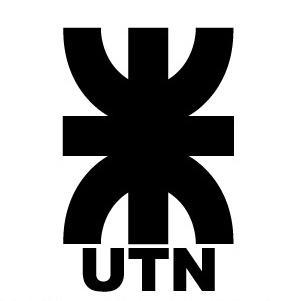
**Comisión: K2.1**

**Ciudad de Resistencia, Chaco**

**Abril - Julio de 2022**

Trabajo Práctico Integrador:

**Entrega Final – Lexer + Parser + Traductor**



Índice

[Introducción 3](#_Toc107797713)

[Objetivo 3](#_Toc107797714)

[Descripción 3](#_Toc107797715)

[Contenido 3](#_Toc107797716)

[Herramientas 3](#_Toc107797717)

[XML (Extensible Markup Language) 3](#_Toc107797718)

[RSS (Really Simple Syndication) 4](#_Toc107797719)

[Python 4](#_Toc107797720)

[Gramática 5](#_Toc107797721)

[Analizador Léxico (Lexer o Scanner) 11](#_Toc107797722)

[Definición de Token, Lexema y Patrón (con ejemplos) 11](#_Toc107797723)

[Definición del AL 12](#_Toc107797724)

[Características del AL 12](#_Toc107797725)

[Procedimiento del AL 12](#_Toc107797726)

[Errores detectables por el AL 12](#_Toc107797727)

[Modo de Obtención del Interprete 13](#_Toc107797728)

[Modo de Ejecución del Interprete 13](#_Toc107797729)

[Código del Lexer 14](#_Toc107797730)

[Ejemplos 18](#_Toc107797731)

[Analizador Sintáctico (Parser) 20](#_Toc107797732)

[Definición 20](#_Toc107797733)

[Cómo funciona un AS 20](#_Toc107797734)

[Errores detectables por el AS 20](#_Toc107797735)

[Modo de Obtención del Intérprete 21](#_Toc107797736)

[Modo de Ejecución del Intérprete 21](#_Toc107797737)

[Código del Parser 22](#_Toc107797738)

[Reglas de gramática del Parser 29](#_Toc107797739)

[Interfaz gráfica 31](#_Toc107797740)

[Conclusión 32](#_Toc107797741)

[Bibliografía o Referencias Web 33](#_Toc107797742)

# Introducción

## Objetivo

En el presente Trabajo Práctico Integrador se pretende desarrollar las siguientes competencias de la asignatura, las cuales son:

Capacidad para reconocer los elementos propios de la Sintaxis y Semántica de los Lenguajes de Programación, pudiendo así comprender los conceptos y procedimientos léxicos/sintácticos asociados, para así poder aplicarlos en la creación de las gramáticas necesarias para detallar los lenguajes que luego se utilizaran en las correspondientes máquinas.

## Descripción

(se actualiza antes de cada entrega)

## Contenido

El documento estará divido en las siguientes partes:

• Primero detallaremos la definición del RSS que vamos a utilizar, declarando cuales son los tokens o los componentes léxicos que acepta (palabras reservadas como terminales) y como deben ser la estructura de las mismas.

• Luego describiremos las producciones de la gramática que utilizara el compilador, detallando los símbolos que utilizara y el significado de cada uno de ellos. El tipo de gramática que utilizamos es libre de contexto.

• Posteriormente profundizaremos con lo que respecta al analizador léxico, mencionando que herramientas utilizamos y una breve descripción de las mismas, en qué consiste el mismo, qué funciones posee, cómo se obtuvo y se ejecuta el intérprete y ejemplos.

## Herramientas

### XML (Extensible Markup Language)

El XML se preocupa por estructurar la información que se pretende almacenar, es decir, define un conjunto de reglas para la codificación de documentos. Este lenguaje de marcado es un conjunto de códigos que se pueden aplicar en el análisis de datos o la lectura de textos creados por computadoras o personas. Se utiliza generalmente para definir elementos, crear un formato y generar un lenguaje personalizado. El diseño XML se centra en la simplicidad, la generalidad y la facilidad de uso y, por lo tanto, se utiliza para varios servicios web, entre otras funcionalidades.

### RSS (Really Simple Syndication)

En simples palabras, es un formato que cumple con el estándar XML.

Este formato es muy estructurado debido a que posee cierto orden o forma.

RSS se usa principalmente para compartir contenido web actualizado desde un sitio web a otros, dado que es un formato estandarizado que facilita la forma de distribuir el contenido y además estos archivos se actualizan automáticamente.

Como todo documento XML se usa para describir, almacenar y transmitir o transportar información. La estructura está marcada por la propia lógica de la información.

### Python

De forma resumida podemos decir que, es un lenguaje de alto nivel de programación (facilidad de utilizar para el programador) interpretado, cuya filosofía hace hincapié en la legibilidad de su código, ya que es lo más parecido al lenguaje humano y se puede aprender de manera sencilla y rápida.

Se utiliza para desarrollar aplicaciones de todo tipo y se trata de un lenguaje de programación multiparadigma (resultado de integrar dos o más paradigmas en un mismo sistema). Es un lenguaje interpretado (no se traduce realmente a un formato legible por el ordenador en tiempo de ejecución), dinámico y multiplataforma (todas las herramientas necesarias están disponibles en todas las plataformas principales).

# Gramática

**Reglas de Producción**

Σ → XML //Llamado XML primero y luego a RSS (obligatoriamente).

XML→ t\_xml RSS //Token (terminal) XML.

RSS→ t\_rss CANAL tc\_rss //Token RSS. Abre y cierra RSS

CANAL→ t\_canal TITULO LINK DESC ITEM tc\_canal **|** t\_canal TITULO LINK DESC OPC ITEM tc\_canal  //Token Channel. /Del canal se va a los 3 elementos obligatorios: Título, Link y Descripción y luego Item./ O: Elementos opcionales.

         TITULO → t\_titulo t\_txt tc\_titulo //título genérico principal del canal.

LINK → t\_link URL tc\_link //link genérico principal del canal.

DESC → t\_desc t\_txt tc\_desc

   OPC → CAT|COPY|IMAGE|LENG|WEBMAST|ULTeDIT||CAT OPC|

// Del canal se va 3 elementos opcionales**:** Category, Copyright, Image. /Puede recursionar hacia el elemento opcional Categoría.

CAT→ t\_cat TXT tc\_cat

COPY→ t\_copy TXT tc\_copy

         IMAGEN→ t\_imagen URL TITULO LINK **|** URL TITULO OPIMAG tc\_imagen //3 elementos obligatorios: Url, Título y Link.

                    OPIMAG→HW | H | W //2 elementos opcionales de la imagen: Width y Height.

                    H→  t\_height N tc\_height

        W→ t\_width N tc\_width

LENG→ t\_leng t\_txt tc\_leng

WEBMAST→ t\_webmast t\_txt tc\_webmast

ULTeDIT→ t\_ult\_edit t\_txt tc\_ult\_edit

ITEM→ t\_item TITULO LINK DESC tc\_item **|** t\_item TITULO LINK DESC tc\_item ITEM // 3 elementos obligatorios en Ítem: Título, Link y Descripción. /Recursión de items.

ITEM→ t\_item TITULO LINK DESC OPI tc\_item **|** t\_item TITULO LINK DESC OPI tc\_item ITEM   // Elementos opcionales de Ítem./ Recursión de ítems con opciones adicionales.

OPI→ CAT **|** COPY **|** AUTOR | FECHA | CAT OPI // El ítem contiene 4 elementos opcionales**:** Category, Copyright, Autor y Fecha. /Puede recursionar hacia el elemento opcional Categoría.

AUTOR→ t\_autor t\_txt tc\_autor

FECHA→ t\_date t\_fecha tc\_date

**Gramática para Links**

URL→ t\_url PRO SUB DOM EXT : PUERTO tc\_url

//del URL pasa al protocolo (parte de su estructura)

PRO→ http:// **|** https:// **|** ftps:// **|** ftp://

//protocolo

○ http:// (para recursos de la web)

○ https:// (para recursos de la web contenidos en un servidor seguro)

○ ftp:// (recursos contenidos en un servidor de ficheros)

○ ftps:// (recursos contenidos en un servidor de ficheros seguro)

PUERTO→ t\_num **|** /DIR  **|** t\_num PUERTO //puerto (opcional)

SUB→ t\_txt. **|** www. **|** ww1 **|**  blog. //subdominio

DOM→ t\_txt/ **|** miweb//dominio

EXT→ t\_txt **|** com **|** ar **|** org**/**/extensión

DIR→ t\_txt **|** t\_txt DIR/ **|** LOC //ruta o directorio (opcional)

LOC→#t\_txt **|** #t\_txt  LOC //localizador interno (opcional)

**Lista de Tokens**

**Lista de los Símbolos No Terminales:**

· XML : Inicio del XML

· RSS : Inicio del RSS

· CANAL : Channel (canal)

. TITULO : Título

. LINK : Link

· DESC : Descripción

· ITEM : Item

· OPC : Opciones del Canal

· CAT : Categoría

· COPY : Copyright

· IMAG : Imagen

· OPIMAG : Opciones de la Imagen

· H : Height

· W : Width

· OPI : Opciones del Ítem

· TXT: Texto Genérico (terminales)

· WWW : Link Genérico (terminales)

· URL : URL

· PRO : Protocolo

· SUB : Subdominio

· DOM : Dominio

· EXT : Extensión

· DIR : Ruta o directorio (opcional)

. LOC : Localizador (opcional)

· PUERTO : Puerto (opcional)

**Lista de los Símbolos No Terminales Adicionales:**

. LENG: Lenguaje del canal

. WEBMAST: El administrador de la página

. ULTeDIT: Última edición de página

. AUTOR: Autor del ítem

. FECHA: Fecha del ítem

**Lista de los Símbolos Terminales/Tokens:**

· t\_xml : <?xml version="t\_num.t\_num" encoding="t\_txt-t\_num"?>

· t\_rss:  <rss version=”t\_num.t\_num”>

· tc\_rss:</rss>

· t\_canal : <channel>

. tc\_canal: </channel>

· t\_titulo : <title>

. tc\_titulo: </title>

· t\_link : <link>

. tc\_link: </link>

· t\_desc : <description>

. tc\_desc : </description>

· t\_cat : <category>

. tc\_cat: </category>

· t\_copy : <copyright>

· tc\_copy : </copyright>

· t\_imag : <image>

· tc\_imag : </image>

· t\_url :  <url>

· tc\_url : </url>

· t\_height : <height>

· tc\_height : </height>

· t\_idth : <width>

. tc\_width : </width>

· t\_item :  <item>

. tc\_item :  </item>

. tc\_date :  </pubDate>

. tc\_date :  </pubDate>

. t\_txt: Texto Génerico

. t\_num: Número Genérico

. t\_fecha: Fecha Genérica

. t\_url: <url>

. tc\_url: </url>

**Lista de los Símbolos Terminales/Tokens Adicionales:**

. t\_language: <language>

. tc\_languaje: </language>

. t\_webMaster: <webMaster>

. tc\_webMaster: </webMaster>

. t\_ultEdit: <lastBuildDate>

. tc\_ultEdit: </lastBuildDate>

. t\_autor: <author>

. tc\_autor: </author>

. t\_fecha: <pubDate>

. tc\_fecha: </pubDate>

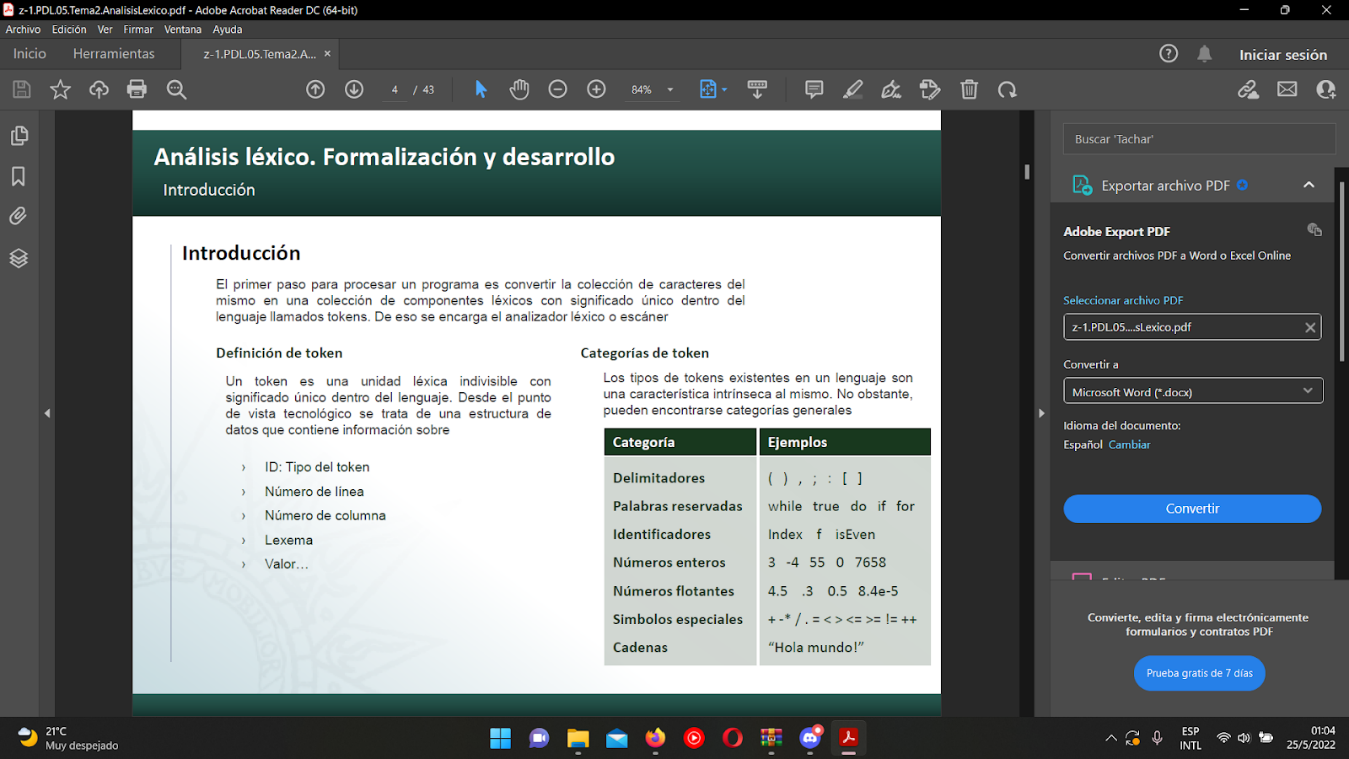
# Analizador Léxico (Lexer o Scanner)

## Definición de Token, Lexema y Patrón (con ejemplos)

Creemos conveniente para una mejor comprensión del tema, dejar una breve definición con nuestra palabras de:

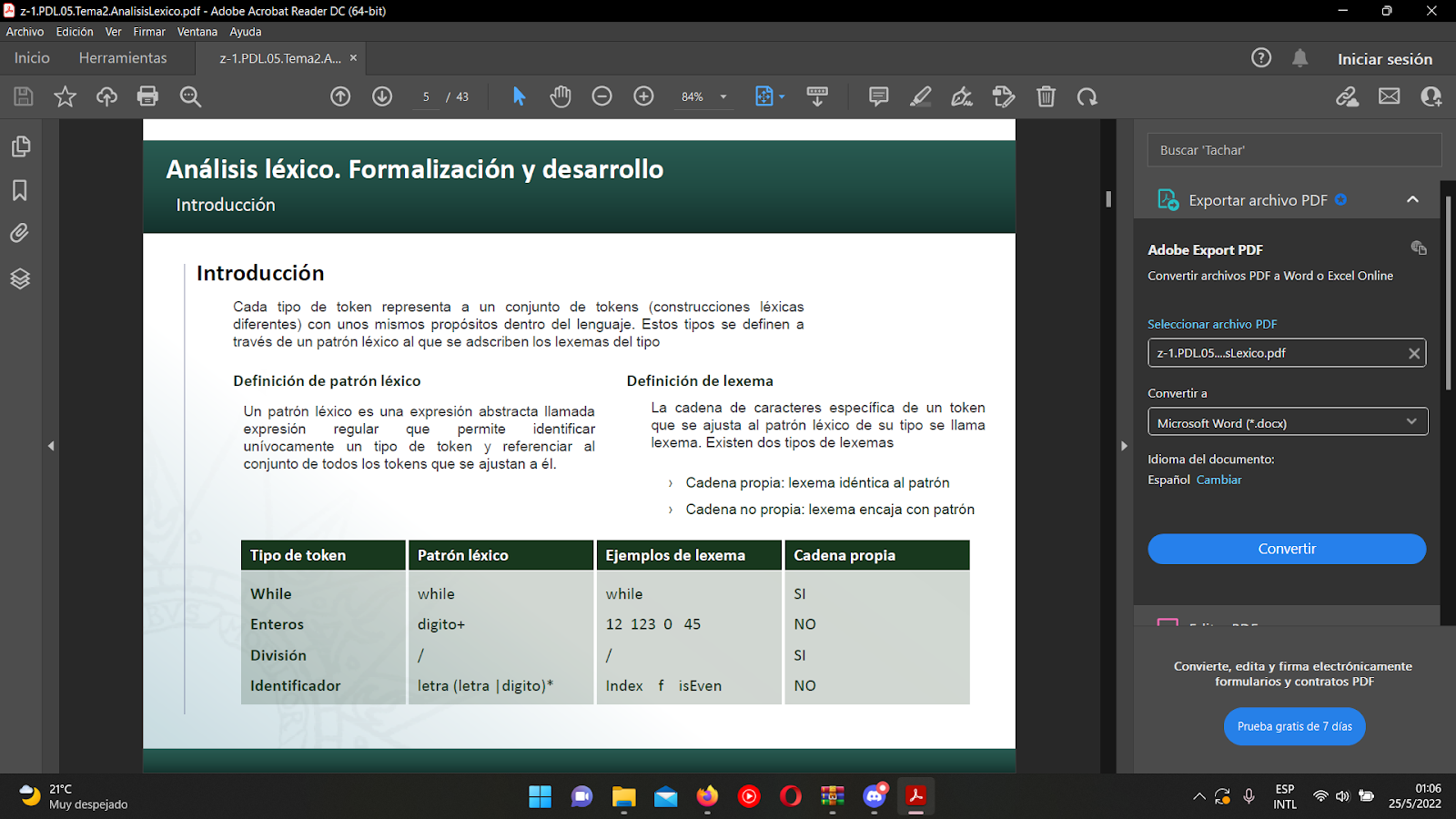
* *Token Léxico o Token:* Es una unidad indivisible con significado único dentro del lenguaje, por lo cual, es la salida del analizador léxico.

Ejemplos de Tokens:



* *Patrón:* Es una expresión regular que permite identificar un tipo de token y referenciar al conjunto de todos los tokens que se ajustan a él.
* 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.

Ejemplo de patrón y lexema:



## Definición del AL

El ***analizador léxico*** es la primera fase un de compilador (programa que traduce código escrito en un lenguaje de programación a otro lenguaje) y lo que hace es recibir como entrada el código fuente de otro programa (secuencia de caracteres) y produce una salida compuesta de tokens (componentes léxicos) o símbolos.

Es decir, es un programa capaz de descomponer una entrada de caracteres (generalmente contenidos en un fichero) en una secuencia ordenada de tokens.

## Características del AL

Por lo cual, las acciones que realiza el lexer son:

* Procesar el lenguaje fuente como una secuencia de caracteres, es decir, convertir la secuencia de caracteres en una colección de componentes léxicos con significado único dentro del lenguaje llamados tokens.
* Agrupar la secuencia de caracteres en palabras con significado propio y después lo transforma en una secuencia de terminales.
* Buscar los componentes léxicos o palabras que componen el programa fuente, según reglas o patrones.

## Procedimiento del AL

1. 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.
2. Se reconocen los tokens, es decir que, los autómatas finitos reconocen los lenguajes por medio de dichas expresiones regulares.
3. 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, números malformados, entre otros.
* Errores de ortografía en palabras reservadas.

Por lo cual se puede decir que detecta etiquetas faltantes o mal escritas y también utilización de símbolos no permitidos.

* Control en el número de caracteres que aparecen en un identificador, por ejemplo, exceder el número de caracteres máximo.
* Fin de archivo a mitad de componente léxico.

## Modo de Obtención del Interprete

Luego de elegir el lenguaje de programación a utilizar (Python), recurrimos a varios videos tutoriales para así poder obtener una librería llamada PLY, la cual nos permitió realizar el analizador léxico. Después de eso, modificamos el formato y añadimos los tokens específicos de la estructura RSS para que sean reconocidos por nuestro Lexer. Posteriormente añadimos funciones específicas para una mejor ejecución del programa.

## Modo de Ejecución del Interprete

Los pasos para la ejecución del interprete son:

1. El usuario ingresará por consola la dirección de la carpeta que contiene los archivos.rss a analizar. Por ejemplo: “C:\Users\Desktop\Lexer\prueba\”.
2. Se interpretarán las sentencias de un archivo pasado como argumento desde la línea de comandos. El archivo deberá tener la extensión “.rss” rss.exe ejemplo.rss.

Logrando que el intérprete:

* Funcione correctamente tanto de forma interactiva como ejecutando las

instrucciones desde los archivos de ejemplo.

* Indique 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).

## Código del Lexer

|  |
| --- |
| #importar librerias import ply.lex as lex import re  import codecs import os import sys import pathlib  reservadas = ['APERTURA\_XML', 'VERSION\_XML', 'ENCODING\_VERSION', 'CIERRE\_XML', 'APERTURA\_RSS', 'VERSION\_RSS', 'CIERRE\_RSS', 'APERTURA\_CHANNEL', 'CIERRE\_CHANNEL', 'APERTURA\_TITLE', 'CIERRE\_TITLE', 'APERTURA\_LINK', 'CIERRE\_LINK', 'APERTURA\_DESC', 'CIERRE\_DESC', 'APERTURA\_CAT', 'CIERRE\_CAT', 'APERTURA\_COPY', 'CIERRE\_COPY', 'APERTURA\_IMAG', 'CIERRE\_IMAG', 'APERTURA\_URL', 'CIERRE\_URL', 'APERTURA\_HEIGHT', 'CIERRE\_HEIGHT', 'APERTURA\_WIDTH', 'CIERRE\_WIDTH', 'APERTURA\_ITEM', 'CIERRE\_ITEM', 'APERTURA\_LANGUAGE', 'CIERRE\_LANGUAGE', 'APERTURA\_WEBMASTER', 'CIERRE\_WEBMASTER', 'URL', 'APERTURA\_ULTEDIT', 'CIERRE\_ULTEDIT', 'APERTURA\_AUTOR','CIERRE\_AUTOR', 'APERTURA\_BD', 'CIERRE\_BD', 'APERTURA\_GUID', 'CIERRE\_GUID', 'APERTURA\_TTL', 'CIERRE\_TTL',  ]  tokens = reservadas+['TXT', 'NUM','IGUAL', 'MENOR','MAYOR','DOBLE\_COMILLA','COMA', 'PUNTO','DOBLE\_PUNTO','DIVISOR', 'ESPACIO', 'MULT','PUNTO\_COMA','PARENT\_I', 'PARENT\_D','PREG\_I','PREG\_D', 'SALTO', 'MAS', 'A','E','I','O','U', 'SANGRIA', 'MENOS', ]  #tokens xml t\_APERTURA\_XML = r'\<\?xml' t\_VERSION\_XML = r'\ version="1.0"' t\_ENCODING\_VERSION= r'\ encoding="UTF-8"' t\_CIERRE\_XML= r'\?\>' #tokens rss t\_APERTURA\_RSS=r'\<rss' t\_VERSION\_RSS=r'\ version="2.0">' t\_CIERRE\_RSS=r'\</rss>' #tokens tags generales t\_APERTURA\_CHANNEL = r'\<channel>' t\_CIERRE\_CHANNEL = r'\</channel>'  t\_APERTURA\_TITLE = r'\<title>' t\_CIERRE\_TITLE = r'\</title>' t\_APERTURA\_LINK=r'\<link>' t\_CIERRE\_LINK=r'\</link>' t\_APERTURA\_DESC=r'\<description>' t\_CIERRE\_DESC=r'\</description>' t\_APERTURA\_CAT=r'\<category>' t\_CIERRE\_CAT=r'\</category>' t\_APERTURA\_COPY=r'\<copyright>' t\_CIERRE\_COPY=r'\</copyright>' t\_APERTURA\_IMAG=r'\<image>' t\_CIERRE\_IMAG=r'\</image>' t\_APERTURA\_URL=r'\<url>' t\_CIERRE\_URL=r'\</url>' t\_APERTURA\_HEIGHT=r'\<height>' t\_CIERRE\_HEIGHT=r'\</height>' t\_APERTURA\_WIDTH=r'\<width>' t\_CIERRE\_WIDTH=r'\</width>' t\_APERTURA\_ITEM=r'\<item>' t\_CIERRE\_ITEM=r'\</item>' t\_APERTURA\_LANGUAGE=r'\<language>' t\_CIERRE\_LANGUAGE=r'\</language>' t\_APERTURA\_WEBMASTER=r'\<webMaster>' t\_CIERRE\_WEBMASTER=r'\</webMaster>' t\_APERTURA\_ULTEDIT=r'\<pubDate>' t\_CIERRE\_ULTEDIT=r'\</pubDate>' t\_APERTURA\_AUTOR=r'\<author>' t\_CIERRE\_AUTOR=r'\</author>' t\_APERTURA\_BD=r'\<lastBuildDate>' t\_CIERRE\_BD=r'\</lastBuildDate>' t\_APERTURA\_GUID=r'\<guid>' t\_CIERRE\_GUID=r'\</guid>' t\_APERTURA\_TTL=r'\<ttl>' t\_CIERRE\_TTL=r'\</ttl>'  t\_ignore = '\t' #definición de tokens simbolos t\_IGUAL = r'=' t\_MENOR = r'<' t\_MAYOR = r'>' t\_DOBLE\_COMILLA = r'"' t\_COMA = r',' t\_PUNTO = r'\.' t\_DOBLE\_PUNTO = r':' t\_DIVISOR = r'/' t\_ESPACIO = r'\ ' t\_MULT = r'\\*' t\_PUNTO\_COMA = r';' t\_PARENT\_I = r'\(' t\_PARENT\_D = r'\)' t\_PREG\_I = r'\¿' t\_PREG\_D = r'\?' t\_MAS = r'\+' t\_MENOS = r'\-' t\_A= r'\á' t\_E= r'\é' t\_I= r'\í' t\_O= r'\ó' t\_U= r'\ú' t\_SALTO= r'\n' t\_SANGRIA= r'\t'  #definición de tokens de cadena de texto def t\_TXT(t):   r'[a-zA-Z][a-zA-Z]\*'  if t.value.upper() in reservadas:  t.value = t.value.upper()  t.type = t.value   return t  #definición de tokens de cadena de numeros def t\_NUM(t):  r'\d+'  t.value = int(t.value)  return t #definición de tokens de cadena de URL def t\_URL(t):  r'http[s]?://(?:[a-zA-Z.0-9/\_=?:#&$-]\*)'  t.type='URL'  return t   #definición de salto de pagina def t\_newline(t):  r'\n+'  t.lexer.lineno += len(t.value)  def t\_COMMENT(t):  r'\#.\*'  pass  def t\_error(t):  print (" LexToken(SALTODEPAG)%s'" % t.value[0])  t.lexer.skip(1)  def buscarFicheros(directorio):  ficheros = []  numArchivo = ''  respuesta = False  cont = 1  #lo que haya dentro del directorio  for base, dirs, files in os.walk(directorio):  ficheros.append(files)   for file in files:  print (str(cont)+". "+file)  cont = cont+1   while respuesta == False:  numArchivo = input('\nIngrese numero de archivo de prueba: ')  for file in files:  if file == files[int(numArchivo)-1]:  respuesta = True  break   return files[int(numArchivo)-1]  directorio = '/home/runner/Lexer/test/' archivo = buscarFicheros(directorio) test = directorio+archivo  #permite leer archivos con tildes fp = codecs.open(test,"r","utf-8")  #lee fp cadena = fp.read() fp.close()  analizador = lex.lex()  analizador.input(cadena)  while True:  tok = analizador.token()  if not tok : break  print (tok) |

## Ejemplos

Captura de pantalla de computadora

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente

# Analizador Sintáctico (Parser)

## Definición

Creemos conveniente para una mejor comprensión del tema, dejar una breve definición:

Un ***analizador sintáctico*** o parser es un programa que normalmente es parte de un compilador. El compilador se asegura de que el código se traduce correctamente a un lenguaje ejecutable. La tarea del analizador es, en este caso, la descomposición y transformación de las entradas en un formato utilizable para su posterior procesamiento. Se analiza una cadena de instrucciones en un lenguaje de programación y luego se descompone en sus componentes individuales.

## Cómo funciona un AS

Para analizar un texto, por ejemplo, los analizadores suelen utilizar un analizador léxico separado (llamado lexer), que descompone los datos de entrada en fichas (símbolos de entrada como palabras). Los Lexers son por lo general máquinas finitas, que siguen la gramática regular y por lo tanto aseguran un desglose adecuado. Los tokens obtenidos de esta manera sirven como caracteres de entrada para el analizador sintáctico.

## Errores detectables por el AS

Un ejemplos de error que detecta nuestro ***Analizador Sintáctico*** es:

* Al intentar realizar la prueba numera 5, nos detecta un error en la línea 7.

Texto

Descripción generada automáticamente

* Si vamos al código que contiene la prueba número 5, podremos comprobar que efectivamente existe un error de sintaxis en la línea 7: No se cerró el Tag de “lastBuildDate”

Texto

Descripción generada automáticamente

## Modo de Obtención del Intérprete

Luego de realizar y corregir el lexer, ya que en la segunda entrega tuvo algunos errores de ejecución, proseguimos con la codificación del parser utilizando la misma librería (PLY) la cual nos permitió realizar el analizador sintáctico. De esta forma, adaptamos y definimos las reglas de gramática en RSS.

## Modo de Ejecución del Intérprete

Para ejecutar el analizador sintáctico, se lo puede ejecutar directamente a través del archivo PARSERV1.exe, y del mismo modo que el leer, debemos ingresar por teclado la dirección de donde se encuentran los archivos de prueba, en la cual luego de realizar el análisis léxico, nos da el resultado de si está correctamente ordenado. Además, como adicional, genera un archivo ‘prueba.html’ donde se traduce los tags RSS a HTML.

## Código del Parser

|  |
| --- |
| import ply.yacc as yacc #importo las mismas librerias que el lexer import os import codecs import re from lex import tokens from sys import stdin import msvcrt   #Sigma def p\_sigma(p):  '''sigma : aperturaxml aperturarss'''  print ("\nFelicidades")  print ("Su codigo es sintacticamente correcto")  #xml def p\_aperturaxml(p):  '''aperturaxml : APERTURA\_XML VERSION\_XML ENCODING\_VERSION CIERRE\_XML'''  print ("TAG Apertura XML")   #rss def p\_aperturarss(p):  '''aperturarss : APERTURA\_RSS VERSION\_RSS canal CIERRE\_RSS'''  print ("TAG Apertura RSS")  #tag CHANNEL def p\_canal\_basico(p):  '''canal : APERTURA\_CHANNEL titulo LINK desc item CIERRE\_CHANNEL'''  print ("TAG channel")   def p\_canal\_basico2(p):  '''canal : APERTURA\_CHANNEL titulo desc LINK item CIERRE\_CHANNEL'''  print ("TAG channel")   def p\_canal\_opcionales(p):  '''canal : APERTURA\_CHANNEL titulo LINK desc opc item CIERRE\_CHANNEL'''  print ("TAG channel")   def p\_canal\_opcionales2(p):  '''canal : APERTURA\_CHANNEL titulo desc LINK opc item CIERRE\_CHANNEL'''  print ("TAG channel")  def p\_item(p):  '''item : APERTURA\_ITEM titulo LINK desc CIERRE\_ITEM'''  print("TAG ITEM")  def p\_item2(p):  '''item : APERTURA\_ITEM titulo desc LINK CIERRE\_ITEM'''  print("TAG ITEM") def p\_item\_recursivo(p):  '''item : APERTURA\_ITEM titulo LINK desc CIERRE\_ITEM item'''  print("TAG ITEM")  def p\_item\_recursivo2(p):  '''item : APERTURA\_ITEM titulo desc LINK CIERRE\_ITEM item'''  print("TAG ITEM")  def p\_item\_opcionales(p):  '''item : APERTURA\_ITEM titulo LINK desc opc CIERRE\_ITEM'''  print("TAG ITEM")  def p\_item\_opcionales2(p):  '''item : APERTURA\_ITEM titulo desc LINK opc CIERRE\_ITEM'''  print("TAG ITEM")  def p\_item\_opcionales\_recursivo(p):  '''item : APERTURA\_ITEM titulo LINK desc opc CIERRE\_ITEM item'''  print("TAG ITEM") def p\_item\_opcionales\_recursivo2(p):  '''item : APERTURA\_ITEM titulo desc LINK opc CIERRE\_ITEM item'''  print("TAG ITEM")  def p\_titulo(p):  '''titulo : APERTURA\_TITLE TXT CIERRE\_TITLE'''  print ("TAG titulo")  def p\_LINK(p):  '''LINK : APERTURA\_LINK URL CIERRE\_LINK'''  print ("TAG link")  def p\_desc(p):  '''desc : APERTURA\_DESC TXT CIERRE\_DESC'''  print ("TAG descripción")  # TAGS OPCIONALES #--------------------------------------------------------------# #TAG CATEGORIA def p\_opc1(p):  '''opc : categoria'''  def p\_categoria(p):  '''categoria : APERTURA\_CAT TXT CIERRE\_CAT'''  print ("TAG categoria")  #TAG COPYRIGHT def p\_opc2(p):  '''opc : copyright'''  def p\_copyright(p):  '''copyright : APERTURA\_COPY TXT CIERRE\_COPY'''  print ("TAG copyright")  #TAG IMAGE def p\_opc3(p):  '''opc : image'''  def p\_image1(p):  '''image : APERTURA\_IMAG URL titulo LINK CIERRE\_IMAG'''  print ("TAG image")  #imagen con elementos opcionales def p\_image2(p):  '''image : APERTURA\_IMAG URL titulo LINK opcimag CIERRE\_IMAG'''  print ("TAG image")  #elementos opcionales de imagen (largo-ancho, largo, ancho) def p\_opcimag1(p):  '''opcimag : height width'''  def p\_opcimag2(p):  '''opcimag : height'''  def p\_opcimag3(p):  '''opcimag : width'''  def p\_height(p):  '''height : APERTURA\_HEIGHT NUM CIERRE\_HEIGHT'''  def p\_width(p):  '''width : APERTURA\_WIDTH NUM CIERRE\_WIDTH'''  def p\_opc4(p):  '''opc : lenguaje'''  def p\_lenguaje(p):  '''lenguaje : APERTURA\_LANGUAGE TXT CIERRE\_LANGUAGE'''  print ("TAG language")  def p\_opc5(p):  '''opc : webmaster'''  def p\_WEBMASTER(p):  '''webmaster : APERTURA\_WEBMASTER TXT CIERRE\_WEBMASTER'''  print ("TAG webmaster")   def p\_opc6(p):  '''opc : ultedit'''  def p\_pubdate(p):  '''ultedit : APERTURA\_ULTEDIT TXT CIERRE\_ULTEDIT'''  print ("TAG pubdate")  def p\_opc7(p):  '''opc : autor'''  def p\_autor(p):  '''autor : APERTURA\_AUTOR TXT CIERRE\_AUTOR'''  print ("TAG author")  def p\_opc8(p):  '''opc : lastbuilddate'''  def p\_lastbuilddate(p):  '''lastbuilddate : APERTURA\_BD TXT CIERRE\_BD'''  print ("TAG lastBuildDate")  def p\_opc9(p):  '''opc : guid'''  def p\_guid(p):  '''guid : APERTURA\_GUID TXT CIERRE\_GUID'''  print ("TAG guid")  def p\_opc10(p):  '''opc : ttl'''  def p\_ttl(p):  '''ttl : APERTURA\_TTL TXT CIERRE\_TTL'''  print ("TAG ttl") # fin elementos opcionales ----------------------------------------#    def p\_error(p):  print ("Error de sintaxis "), p  print ("Error en la linea "+str(p.lineno))  def buscarFicheros(directorio):  ficheros = []  numArchivo = ''  respuesta = False  cont = 1   for base, dirs, files in os.walk(directorio):  ficheros.append(files)   for file in files:  print (str(cont)+". "+file)  cont = cont+1   while respuesta == False:  numArchivo = input('\nNumero del test: ')  for file in files:  if file == files[int(numArchivo)-1]:  respuesta = True  break   print ("Tags reconocidos en el archivo \"%s\" \n" %files[int(numArchivo)-1])   return files[int(numArchivo)-1]  print ("analizador sintactico") print ("Por favor, ingrese por teclado la dirección de la carpeta en donde se encuntra los archivos.rss") directorio = input() +'/' archivo = buscarFicheros(directorio) test = directorio+archivo fp = codecs.open(test,"r","utf-8") cadena = fp.read() fp.close()  parser = yacc.yacc() result = parser.parse(cadena)   print (result)   # archivo input rss  fin = open(test) # archivo output html fout = open(("prueba.html"), "wt") # iteracion buffer = fin.read()  # reemplazo buffer = buffer.replace('<title>', '<h1>') buffer = buffer.replace('</title>', '</h1>') buffer = buffer.replace('<description>', '<p>') buffer = buffer.replace('</description>', '</p>') buffer = buffer.replace('<link>', '<a>') buffer = buffer.replace('</link>', '</a>') buffer = buffer.replace('<url>', '<a>') buffer = buffer.replace('</url>', '</a>')  fout.write(buffer) fin.close() fout.close()    msvcrt.getch() |

## Reglas de gramática del Parser

Rule 0     S' -> sigma

Rule 1     sigma -> aperturaxml aperturarss

Rule 2     aperturaxml -> APERTURA\_XML VERSION\_XML ENCODING\_VERSION CIERRE\_XML

Rule 3     aperturarss -> APERTURA\_RSS VERSION\_RSS canal CIERRE\_RSS

Rule 4     canal -> APERTURA\_CHANNEL titulo LINK desc item CIERRE\_CHANNEL

Rule 5     canal -> APERTURA\_CHANNEL titulo LINK desc opc item CIERRE\_CHANNEL

Rule 6     item -> APERTURA\_ITEM titulo LINK desc CIERRE\_ITEM

Rule 7     item -> APERTURA\_ITEM titulo LINK desc CIERRE\_ITEM item

Rule 8     item -> APERTURA\_ITEM titulo LINK desc opc CIERRE\_ITEM

Rule 9     item -> APERTURA\_ITEM titulo LINK desc opc CIERRE\_ITEM item

Rule 10    titulo -> APERTURA\_TITLE TXT CIERRE\_TITLE

Rule 11    LINK -> APERTURA\_LINK URL CIERRE\_LINK

Rule 12    desc -> APERTURA\_DESC TXT CIERRE\_DESC

Rule 13    opc -> categoria

Rule 14    categoria -> APERTURA\_CAT TXT CIERRE\_CAT

Rule 15    opc -> copyright

Rule 16    copyright -> APERTURA\_COPY TXT CIERRE\_COPY

Rule 17    opc -> image

Rule 18    image -> APERTURA\_IMAG URL titulo LINK CIERRE\_IMAG

Rule 19    image -> APERTURA\_IMAG URL titulo LINK opcimag CIERRE\_IMAG

Rule 20    opcimag -> height width

Rule 21    opcimag -> height

Rule 22    opcimag -> width

Rule 23    height -> APERTURA\_HEIGHT NUM CIERRE\_HEIGHT

Rule 24    width -> APERTURA\_WIDTH NUM CIERRE\_WIDTH

Rule 25    opc -> lenguaje

Rule 26    lenguaje -> APERTURA\_LANGUAGE TXT CIERRE\_LANGUAGE

Rule 27    opc -> webmaster

Rule 28    webmaster -> APERTURA\_WEBMASTER TXT CIERRE\_WEBMASTER

Rule 29    opc -> ultedit

Rule 30    ultedit -> APERTURA\_ULTEDIT TXT CIERRE\_ULTEDIT

Rule 31    opc -> autor

Rule 32    autor -> APERTURA\_AUTOR TXT CIERRE\_AUTOR

Rule 33    opc -> lastbuilddate

Rule 34    lastbuilddate -> APERTURA\_BD TXT CIERRE\_BD

Rule 35    opc -> guid

Rule 36    guid -> APERTURA\_GUID TXT CIERRE\_GUID

Rule 37    opc -> ttl

Rule 38    ttl -> APERTURA\_TTL TXT CIERRE\_TTL

# Interfaz gráfica

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Como adicional en esta entrega, decidimos implementar una simple interfaz gráfica ejecutable, que funciona como acceso directo de ambos archivos (LEXERV1.exe y PARSERV1.exe). La misma se encuentra en la carpeta ‘’bin’.

# Conclusión

En la primera entrega, aprendimos a aplicar el diseño de gramáticas visto en clase a un ejemplo concreto, en este caso, el del formato RSS. Este formato nos resultó sencillo de entenderlo, ya que todos los integrantes del grupo teníamos, al menos un pequeño conocimiento previo del mismo. No tuvimos grandes dificultades a la hora de diseñar la gramática, salvo en la aplicación de la recursión en las producciones gramaticales y de organizar un horario en común para dedicarnos al desarrollo del trabajo práctico.

En la segunda entrega aprendimos a diseñar un analizador léxico mediante un lenguaje de programación, en este caso Python, ya que nos pareció el más práctico al momento de realizar la codificación. En el proceso aprendimos a utilizar dicho lenguaje, ya que la mayoría de integrantes del grupo no sabían utilizarlo y fue una experiencia muy satisfactoria ya que pudimos observar la aplicación de lo aprendido en clase ese momento. Algunas complicaciones que se nos presentaron fueron con respecto a las instalaciones de las diferentes librerías a utilizar, como así también errores de sintaxis con respecto a algunas líneas del código. Para solucionar esto recurrimos a diferentes fuentes y vídeos en donde pudimos corregir dichos problemas. Un problema puntual que se nos presentó y no supimos cómo solucionarlo es el de obtener el directorio donde se encuentran los archivos .rss de prueba automáticamente, por lo que en el programa dicho directorio se debe ingresar manualmente.

A lo largo de la construcción de este trabajo final, se pudo poner en práctica lo aprendido este cuatrimestre. Empezando con la realización de una gramática LDC hasta implementar todo para la construcción de un compilador.

En el proceso de codificación tuvimos varios inconvenientes, debido a no tener mucha experiencia previa Py. Por lo anterior mencionado hemos perdido mucho tiempo en investigar cómo funcionaba el lenguaje.

Luego de muchas pruebas y errores pudimos realizar el proyecto, lo cual se creyó que sería imposible.

La realización de este proyecto nos permitió comprender el funcionamiento de los analizadores e intérpretes, logrando así ampliar nuestro conocimiento en programación/desarrollo de software y familiarizarnos aún más con ello.

# Bibliografía o Referencias Web

Wikipedia - Comparison of parser generators: <https://en.wikipedia.org/wiki/Comparison_of_parser_generators>

YouTube - ¿Qué es HTML? - 10 cosas que debes saber:

https://www.youtube.com/watch?v=tPzq8IufGxE

YouTube - Curso Básico de HTML desde 0 – Introducción:

https://www.youtube.com/watch?v=cqMfPS8jPys&list=PLhSj3UTs2\_yVHt2DgHky\_MzzRC58UHE4z

YouTube - ¿Qué es RSS?:

https://www.youtube.com/watch?v=t5m5lKx6rEo vídeo de qué es RSS

W3 Schools - XML RSS:

https://www.w3schools.com/xml/xml\_rss.asp

Youtube - Tutorial #Como crear un RSS:

https://www.youtube.com/watch?v=0uNZDzi\_jYs

W3C - W3C Feed Validation Service3:

https://validator.w3.org/feed/

YouTube - ¿QUÉ ES XML Y PARA QUÉ SE USA?:

https://www.youtube.com/watch?v=AZihBEg8VBk

YouTube - What is XML | XML Beginner Tutorial | Learn XML with Demo in 10 min:

https://www.youtube.com/watch?v=1JblVElt5K0

YouTube - Introducción a XML - 1 - Tutorial XML básico en español:

https://www.youtube.com/watch?v=PxGICnkFZJU&list=PLM-p96nOrGcYb96AMy3VdUN8fo-sVAx7K

YouTube - How to Create Simple XML Document:

https://www.youtube.com/watch?v=i9E\_mOptuEo

Landofcode - RSS tags reference:

<http://www.landofcode.com/rss-reference/rss-tags.php>

https://youtu.be/TG0qRDrUPpA Qué es el Lexer, para que sirve, cómo empezar, ejemplos de lexer

https://www.youtube.com/watch?v=gWrmCOTrtrs Lexer en Python

<https://programmerclick.com/article/4860986790/> Escribir Analizador Léxico a mano

<https://programmerclick.com/article/4860986790/> AL en Python

<http://acodigo.blogspot.com/2014/02/analizador-lexico.html> Analizador Léxico en C

<https://www.youtube.com/watch?v=cRvmSMWLZlY>

<https://es.wikipedia.org/wiki/Lex_(inform%C3%A1tica)> YACC Analizador Léxico Programa en C

<https://youtu.be/Hh49BXmHxX8> Calculadora en PY (analizador léxico)

<https://youtu.be/orI232lQv6U> Calculadora en PY (analizador léxico)

<https://youtu.be/Zbk0lic04SI> Calculadora en PY (analizador léxico)

<https://youtu.be/DOuWQNln9wc> Calculadora en PY (analizador léxico)

<https://youtu.be/WrYTi90ey0E> Calculadora en PY (analizador léxico)