Universidad Tecnológica Nacional

Facultad Regional Resistencia

Logotipo

Descripción generada automáticamente

Ingeniería en Sistemas de Información

Sintaxis y Semántica del Lenguaje

Trabajo Práctico Integrador

Diseño e implementación de Lexer y Parser

Integrantes:

* Ivan, Niveyro
* Juan Pablo, Bravo

1er Cuatrimestre

Curso académico: 2022

Lugar y Fecha:

* Resistencia. Chaco
* 14 de abril

*Índice*

[Introducción 3](#_Toc104662040)

[Información y requerimientos para la ejecución del tp 3](#_Toc104662041)

[Características del lenguaje RSS 3](#_Toc104662042)

[Componentes léxicos 3](#_Toc104662043)

[Sentencias 4](#_Toc104662044)

[Gramática 5](#_Toc104662045)

[Análisis Léxico 7](#_Toc104662046)

[Modo de Obtención del intérprete 7](#_Toc104662047)

[Modo de Ejecución del intérprete 7](#_Toc104662048)

[Iterativa 7](#_Toc104662049)

[A partir de un archivo 7](#_Toc104662050)

[Ejemplos 8](#_Toc104662051)

[Conclusión 8](#_Toc104662052)

[Bibliografía o referencias web 8](#_Toc104662053)

# Introducción

Representamos el grupo 4, el cual está integrado por Iván Niveyro y Bravo Juan Pablo, somos estudiantes de 2do año de la Universidad Tecnológica Nacional en la sede de Resistencia (UTNFrre).

El siguiente trabajo está supervisado por el Profesor Nico Tortosa y tiene como objetivo desarrollar e integrar conceptos para el desarrollo de un compilador, cuyo procedimiento será explicado en el propio documento.

Primero debemos decir que para la realización tuvimos que consultarnos de algunos videos de la plataforma YouTube, algunas páginas web para la obtención de información, consultas al profesor supervisor, todo esto para obtener la suficiente inspiración y para poder ir desarrollando el código o solucionando pequeños problemas que se iban presentando. El desarrollo de nuestra gramática no nos dio demasiados problemas, aunque tuvimos que ir haciendo algunos pequeños retoques para mejorarla. Nuestro compilador decidimos programarlo en el lenguaje Pyhton v.3.10 incorporando la librería externa PLY para la creación del lexer y el parser.

## Información y requerimientos para la ejecución del tp

* Se lo debe ejecutar en sistema operativo Windows.
* La versión de Python debe ser 3.10

# Características del lenguaje RSS

## Componentes léxicos

'XMLCIERRE',

'XMLINICIO',

'COMILLAS',

'IGUAL',

'COMILLASDOBLES',

'COMILLASSIMPLES',

'SALTOLINEA',

'NUMERO',

'PUNTO',

'CERRADURA',

'GUION','ENCODING',

'ITEMINICIO',

'ITEMCIERRE',

'ID','VERSION',

'RSSAPERTURA',

'DESCRIPTIONAPAERTURA',

'CHANNELAPERTURA',

'ITEMAPERTURA',

'LINKAPERTURA',

'TITLEAPERTURA',

'RSSCERRADURA',

'DESCRIPTIONCERRADURA',

'CHANNELCERRDAURA',

'ITEMCERRADURA',

'LINKCERRADURA',

'TITLECERRADURA','IGUAL',

'HTMLS','HTML','FPT','FPTS',

'DOSPUNTOS','BARRA'

## Sentencias

Definiciones de los tokens

# Gramática

Terminales: sin negrita

No terminales: mayúscula, negrita

**Σ** → **XML RSS CHANNEL** </rss>

**XML** →<?xml **VERSION** ?>

**VERSION** →version="1.0"

**VERSION** →version="1.0" **ENCODING**

**ENCODING** →encoding="**CODIF**"

**CODIF** →ISO-8859-1

**CODIF** → UTF-8

**CODIF** →UTF-16

**CODIF** →UTF-32

**RSS** → <rssversion="2.0">

**CHANNEL** → <Channel> **CONTENIDO ITEM** </Channel>

**CONTENIDO** → **TITLE LINK DESCRIPTION**

**CONTENIDO** → **TITLE LINK DESCRIPTION OPCIONALES**

**ITEM** → <Item> **CONT\_ITEM** </Item> **ITEM**

**ITEM** →<Item> **CONT\_ITEM** </Item>

**CONT\_ITEM** → **TITLE LINK DESCRIPTION CATEGORY**

**CONT\_ITEM** → **TITLE LINK DESCRIPTION**

**OPCIONALES → CATEGORY COPYRIGHT IMAGE**

**OPCIONALES → CATEGORY IMAGE**

**OPCIONALES → CATEGORY COPYRIGHT**

**OPCIONALES → CATEGORY**

**OPCIONALES → COPYRIGHT IMAGE**

**OPCIONALES → COPYRIGHT**

**OPCIONALES → IMAGE**

**CATEGORY →** letra **CATEGORY**

**CATEGORY →** letra

**IMAGE** **→ URL TITLE LINK**

**IMAGE** **→ URL TITLE LINK N**

**N** **→ HEIGHT WIDHT**

**HEIGHT →** hmax

**WIDHT →** wmax

**COPYRIGHT →** copyright **COPYRIGHT**

**COPYRIGHT →** copyright

**TITLE** → <title> **E** </title>

**LINK** → <link> **URL** </link>

**DESCRIPCIÓN** → <description> **E** </description>

**E** → texto **E**

**E** → texto

**URL** → **PROTOCOLO** :// **NDOM** : **PUERTO** / **RUTA** # **FRAGMENTO**

**PROTOCOLO** → http

**PROTOCOLO** → https

**PROTOCOLO** → ftp

**PROTOCOLO** → ftps

**NDOM** → texto **NDOM**

**NDOM** → texto

**PUERTO** →80

Comentario: Cualquier otro puerto

**PUERTO** → **E**

**RUTA** → /index.html

**RUTA** → /**E**/**RUTA**

**RUTA** → /**E**

**FRAGMENTO** → **E**

El terminal texto lo tomamos como que incluye a los siguientes caracteres:

texto: (“a”...”z”), (“0”...”9”), (“A”...”Z”), (“á”...”ú”), {“.”, “/”, “?”, “=”, “:”, “,”, “ ”}

letra: (“a”...”z”), (“0”...”9”), (“A”...”Z”), (“á”...”ú”), { “:”, “,”, “ ”}

copyright: (“a”...”z”), (“0”...”9”), (“A”...”Z”), (“á”...”ú”), {“.”, “/”, “:”, “,”, “ ”, “©”}

hmax: (“1”...”400”)

wmax: (“1”...”144”)

# Análisis Léxico

Nombre del Archivo: -----

El analizador léxico recibe como entrada el código fuente de un programa escrito en el lenguaje de programación RSS y produce una salida la cual está compuesta por los tokens definidos, y que nos servirán para el analizador sintáctico.

Básicamente se encarga identificar que las palabras reservadas, caracteres especiales y los caracteres ilegales se detecten, además de indicar la línea y posición en el que está.

# Análisis Sintáctico

# Modo de Obtención del intérprete

# Modo de Ejecución del intérprete

Para la ejecución del interprete primeramente hay que situarse en la carpeta (“Analizador”) y abrir una terminal en esa ubicación, una vez abierta la terminal, se debe escribir el siguiente comando:

Para ejecutar el Analizador Léxico. Ahí se darán dos opciones de ingreso, por consola o por un archivo previamente guardado en la carpeta (“Analizador -> Test”) donde se debe colocar en formato de texto plano los archivos que quieran analizar.

## Iterativa

- Se podrá realizar la prueba correspondiente, una vez que se ejecute el analizador desde la consola del sistema, escribiendo la cadena deseada a analizar.

## A partir de un archivo

- Se podrá realizar la prueba correspondiente a partir de un archivo tomado como entrada. Por ejemplo, un archivo .txt donde estará escrita la cadena y así el programa realizará el análisis correspondiente.

# Ejemplos

# Conclusión

# Bibliografía o referencias web

* <https://docs.microsoft.com/es-es/dotnet/standard/base-types/character-classes-in-regular-expressions>
* <https://www.w3schools.com/xml/xml_rss.asp>
* <http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/guia_rss.htm>