Trabajo Práctico Integrador

- Grupo número: 6
- Integrantes:
  - Bella Matias Nicolas.
  - Molinas González Víctor.
  - Lezcano Claudio Federico.
  - Gomes Martin Maximiliano.
- Asignatura: Sintaxis y semántica de los lenguajes.
- Carrera: Ingeniería de Sistemas de Información.
- Primer cuatrimestre.
- Ciclo académico: 2022.
- Universidad Tecnológica Nacional facultad regional de Resistencia.

Índice

Introducción

El trabajo documentado a continuación consiste en el desarrollo de un analizador léxico y sintáctico para la validación de contenido distribuido en el formato web *RSS*.

RSS es un formato de sindicación de contenido web derivado de XML, su propósito es facilitar la distribución de contendido actualizado de forma frecuente mediante un sistema de suscripciones y sin la necesidad de acceder al sitio web o blog en sí mismo.

Se trabajará a lo largo de tres etapas principales:

1. Desarrollo de la gramática

Se utilizaron conocimientos de teoría de lenguajes y gramáticas formales para formular un conjunto de producciones, estas comprenderán la gramática a modo de una descripción formal de RSS.

En esta primer etapa fué crítico, como primera instancia, evaluar la complejidad que se requería de las producciones gramaticales para establecer un conjunto de reglas coherentes y fáciles de entender; éstas deben permitir generar sólamente contenido válido en formato RSS a la vez que evitar complejidades que podrían afectar su interpretación y, potencialmente, el desempeño del parser.

Analizando el formato característico de los documentos RSS y teniendo en cuenta la jerarquía de lenguajes de Chomsky se optó por una gramática de tipo 2, libre de contexto, para la formulación de las producciones.

2. Desarrollo del lexer

La segunda etapa consistió en la conformación de un lexer que deberá ser capaz de identificar cada uno de los tokens dentro del documento, este conformará la primer etapa de detección de errores en el contenido, ya sean lexicos o sintácticos.

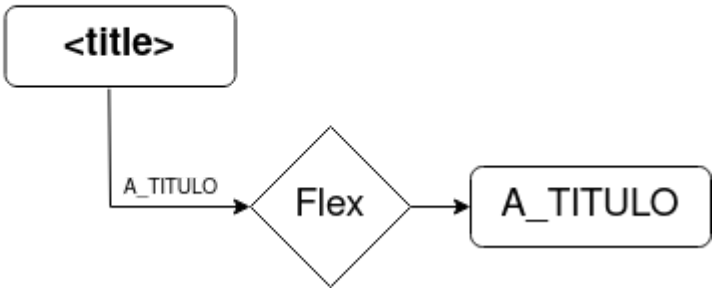
Para su elaboración se dependió de la herramienta Flex cuya función es la de generación de escáners léxicos a través de una descripción que funciona a modo de *esqueleto* para el escáner, la cual puede especificarse mediante un archivo con extensión `.l` o simplemente a través de entrada estándar.

El paso siguiente a la elaboración de la descripción es la producción del código fuente del scanner ejecutando flex y pasándole como parámetro la descripción de los tokens y las reglas, ya sea en la forma de un archivo o mediante entrada estándar. Este paso generará un archivo, por defecto `lex.yy.c`, donde se definen las funciones de control del escáner como `yylex`, y tokens especiales de uso interno como `YYEOF` que es usado para parar la ejecución una vez se ha llegado al final del archivo de entrada cuando se ha declarado la opción `%option noyywrap`.

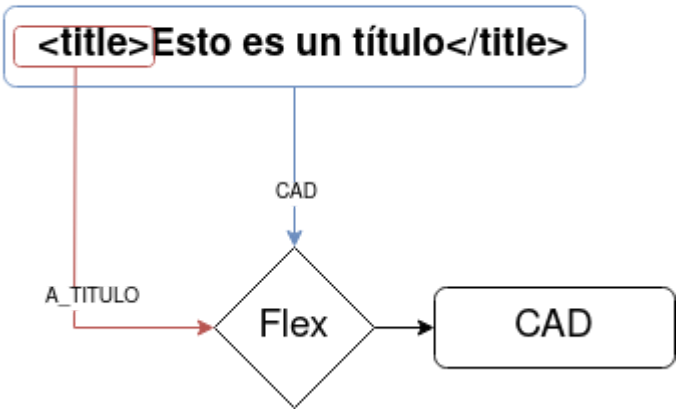
Inconvenientes en la implementación del lenguaje

Si bien la elaboración de la descripción del escáner resultó ser sencilla se encontró que el comportamiento del analizador léxico resultante produjo la imposibilidad de incluir en el alfabeto los símbolos `<` y `>` por situaciones como la siguiente:

Considerando los tags de apertura y clausura de título mientras la apertura esté aislada el analizador los reconocerá sin problemas.



Pero en el momento en que haya un conflicto con otro posible token el comportamiento por defecto es tomar el que abarque la mayor cantidad de texto de la entrada:



Imposibilitando la inclusión de los símbolos en el alfabeto.

3. Desarrollo del parser

El desarrollo del parser presentó varias similitudes con el proceso de conformación del lexer, nuevamente fue necesario confeccionar una descripción, en este caso de la gramática a la que el lenguaje debe adherirse, a través de un archivo con extensión `.y` o mediante entrada estándar.

Para su confección se tuvieron que considerar aspectos como:

- La inclusión de `lex.yy.c` para que el parser tenga acceso a la función `yylex` de control del analizador sintáctico para hacer peticiones de tokens.
- La especificación de un parser LR generalizado debido a la ambigüedad de la gramática mediante la opcion `%glr-parser` que permite que actúe como un aceptor pushdown pudiendo hacer backtracking cuando una derivación es errónea.
- La inclusión de declaraciones `%dprec n` para establecer la precedencia entre formas sentenciales cuando las reglas eran ambiguas.

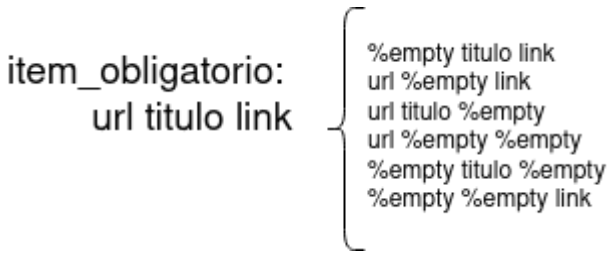
- La inclusión de la función de control `main()` encargada de llamar a la función de control de parser `yyparse()` que, a su ves, hará las llamadas al lexer mediante la función `yylex()`.

Inconvenientes en la elaboración de la gramática

Una de las desventajas de trabajar con gramáticas de tipo 2 es que cada una de las permutaciones en todas las reglas debe ser considerada y ser expresada por separado. Esta fue la principal razón de que en la gramática no se haya podido considerar todas las producciones posibles de canal, item e imagen, ya que la cantidad de producciones eran $7!$, $4!$ y $5!$ respectivamente, requiriendo tiempo del que no se disponía.

La alternativa por la que se optó es agrupar, en el caso del canal por ejemplo, los tags obligatorios y opcionales por separado y considerar todas las permutaciones ($3!$) en cada una de esas agrupaciones. Lo mismo se llevó acabo con la regla de item e imágen.

Otro artilugio que se empleó para facilitar la elaboración de la gramática es la derivación de no terminales en el string vacío mediante la regla `%empty` ya que de esta forma fué posible obviar las producciones donde se realiza un cambio de no terminal o donde deriva en dos no terminales.



Requerimientos de software

Para compilar es necesario tener instalado:

- `bison 3.8.2`
- `flex 2.6.4`
- `gcc 12.1.1`

Opcionalmente:

- `GNU make 4.3`

Instrucciones de compilación

Simplemente ejecutar `make` sobre el directorio raíz o:

```
flex src/lex.l
bison -d src/parser.y
gcc src/parser.tab.c -o bin/parserRSS.sh
```

Gramática

Consideraciones:

- Los no terminales estarán encerrados entre llaves.

Tipos de datos y derivados

Números enteros

```
{NUM}        --> 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
{VERSION}    --> {NUM} | {NUM}.{VERSION}
{ALTO}        --> <height>{NUM}</height>
{ANCHO}        --> <width>{NUM}</width>
```

Cadena de caracteres

```
{CAD}        -->  \b | \t | \n | a | b | c | d | e | f | g | h | i | j | k | l
| m | n | o | p | q | r | s | t | u | v | w | x | y | z | 1 | 2 | 3 | 4 | 5 | 6
| 7 | 8 | 9 | 0 | , | . | ¿ | ? | ¡ | ! | / | \ | $ | ` | ´ | " | # | % | @ | &
| - | _

{CAD}        --> A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P
| Q | R | S | T | U | V | W | X | Y | Z

{TITULO}    --> <title>{CAD}</title>
{DESC}        --> <description>{CAD}<description>
{CATEGORÍA}    --> <category>{CAD}</category>
{DERECHOS}    --> <copyright>{CAD}</copyright>
```

URL

```
{PROT}        --> http | https | ftp | ftps

{CAR}        --> a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p
| q | r | s | t | u | v | w | x | y | z | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
| _ | - | & | ? | =

{CAR}        --> A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P
| Q | R | S | T | U | V | W | X | Y | Z

{RUTA}        --> {CAR} | {CAR}/{RUTA}
{DOMINIO}    --> {CAR} | {CAR}.{DOMINIO}
{PUERTO}    --> {NUM}

{ENLACE}    --> {PROT}:// | {PROT}://{DOMINIO}/ | {PROT}://{DOMINIO}/{RUTA} |
{PROT}://{DOMINIO}:{PUERTO}/ | {PROT}://{DOMINIO}:{PUERTO}/{RUTA}#{CAR} |
{PROT}://{DOMINIO}/{RUTA}#{CAR}

{LINK}        --> <link>{ENLACE}</link>
{URL}        --> <url>{ENLACE}</url>
```

Cuerpo del documento

Cabecera

```
{SIGMA}        --> {DEFXML}{DEFRSS}

{VERSIONADO}    --> version
{CODIFICACION}    --> encoding
{IGUAL}        --> =

{DEFXML}    --> <?xml{VERSIONADO}{IGUAL}"{VERSION}"{CODIFICACION}{IGUAL}"
{CAD}"?> | <?xml{VERSIONADO}{IGUAL}"{VERSION}”?>
{DEFRSS}    --> <rss{VERSIONADO}{IGUAL}"{VERSION}">{CANAL}</rss>
```

Canal

```
{CANAL-OBLIGATORIO}    --> {TITULO}{LINK}{DESC}

{CANAL-OPCIONAL}    --> {CATEGORIA}{DERECHOS}{IMAGEN} |  {CATEGORIA}{DERECHOS}
| {CATEGORIA}{IMAGEN} | {DERECHOS}{IMAGEN} | {CATEGORIA} | {DERECHOS} |
{IMAGEN}

{CANAL-VACIO}        --> <channel>{CANAL-OBLIGATORIO}</channel> | <channel>
{CANAL-OBLIGATORIO}{CANAL-OPCIONAL}</channel>

{CANAL-ITEMS}        --> <channel>{CANAL-OBLIGATORIO}{ITEMS}</channel> |
<channel>{CANAL-OBLIGATORIO}{CANAL-OPCIONAL}{ITEMS}</channel>

{CANAL}            --> {CANAL-VACIO} | {CANAL-ITEMS}
```

Items

```
{ITEM-OBLIGATORIO}    --> {TITULO}{LINK}{DES} | {TITULO}{DESC}{LIN} | {LINK}
{TITULO}{DES} | {LINK}{DESC}{TITUL} | {DESC}{LINK}{TITUL} | {DESC}{TITULO}{LIN}

{ITEM}            --> <item>{ITEM-OBLIGATORIO}</item> | <item>{CATEGORIA}{ITEM-
OBLIGATORIO}</item> | <item>{ITEM-OBLIGATORIO}{CATEGORIA}</item>

{ITEMS}            --> {ITEM}{ITEMS} | {ITEM}
```

Imágen

```
{IMAGEN-OBLIGATORIO}    --> {TITULO}{URL}{LINK} | {TITULO}{LINK}{URL} | {LINK}
{TITULO}{URL} | {LINK}{URL}{TITULO} | {URL}{TITULO}{LINK} | {URL}{LINK}TITULO}

{IMAGEN-OPCIONAL}    --> {ALTO}{ANCHO} | {ANCHO}{ALTO} | {ALTO} | {ANCHO}

{IMAGEN}        --> <image>{IMAGEN-OBLIGATORIO}</image> | <image>{IMAGEN-
OBLIGATORIO}{IMAGEN-OPCIONAL}</image> | <image>{IMAGEN-OPCIONAL}{IMAGEN-
OBLIGATORIO}</image>
```

Símbolos no terminales

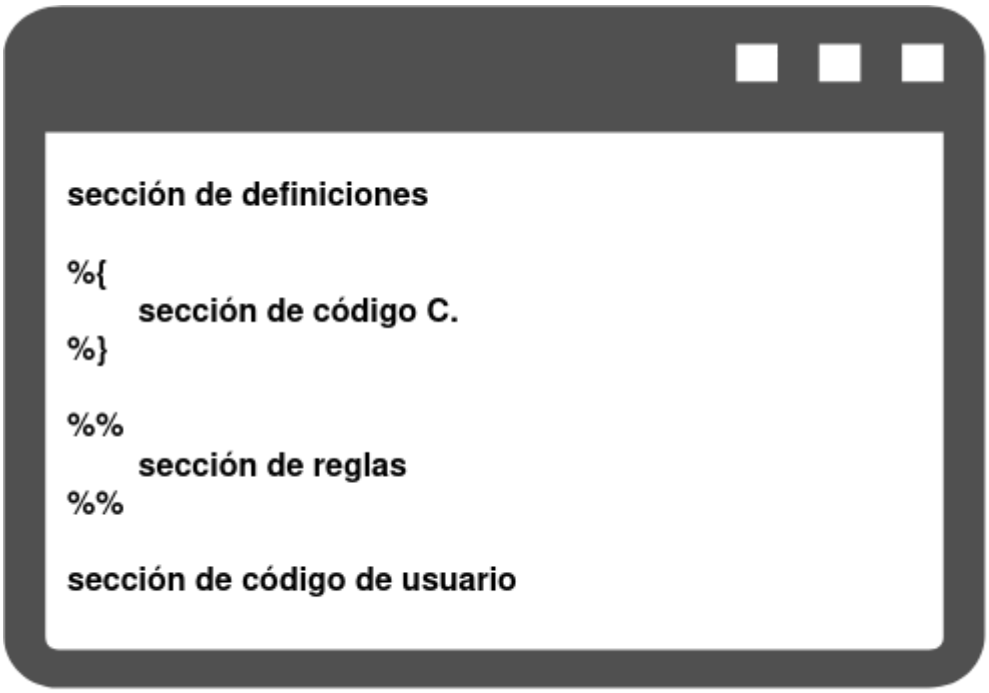
```
{NUM}
{CAD}
{TÍTULO}
{DESC}
{CATEGORÍA}
{DERECHOS}
{PROT}
{CAR}
{LINK}
{URL}
{DEFXML}
{DEFRSS}
{CANAL-OBLIGATORIO}
{CANAL-OPCIONAL}
{CANAL-VACIO}
{CANAL-ITEMS>}
{ITEM}
{ITEMS}
{ITEM-OBLIGATORIO}
{ALTURA}
{ANCHO}
{IMAGEN-OBLIGATORIO}
{IMAGEN-OPCIONAL}
{IMAGEN}
{DOMINIO}
{VERSION}
{RUTA}
{VERSIONADO}
{CODIFICACION}
{IGUAL}
```

Símbolos terminales

```
1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 0 - a - b - c - d - f - g - h - i - j - k -
l - m - n - o - p - q - r - s - t - u - v - w - x - y - z - A - B - C - D - E -
F - G - H - I - J - K - L - M - N - O - P - Q - R - S - T - U - V - W - X - Y -
Z - , -. - ¿ - ? - ¡ - ! - / - \ - $ - ` - ´ - " - # - % - @ - & - _ - - - \
b - \t - \n - <title> - </title> - <description> - <description> - <category> -
</category> - <?xml - ?> - <rss - > - </rss> - <channel> - </channel> - <item>
- </item> - <copyright> - </copyright> -
</image> - <image> - </width> - <width> - http - https - ftp - ftps
```

Lexer

La descripción del escáner estará compuesta de cuatro partes:



Sección de definiciones

Contiene declaraciones de definiciones nombradas para simplificar las especificaciones del escáner y declaraciones de condiciones de inicio.

Una definición es una palabra que compienza con letra o guión bajo y que puede contener letras, números, guión bajo o medio y que está precedida por una expresión regular de la cual funciona como referencia.

Ej:

```
DIGITO       [0-9]
```

Sección de código C

Nos permite definir macros, importar librerías, etc. que podrán ser útiles en la sección de reglas cuando queramos retornar algún valor o imprimir a salida estándar algún resultado.

Todo lo que esté presente en esta sección será copiado literalmente al archivo `lex.yy.c`.

Sección de reglas

Esta sección contiene una serie de reglas que siguen el siguiente patrón:

```
patrón        acción
```

Donde:

- `patrón` : puede ser una referencia a una expresión regular o una expresión en sí mismo y no puede estar indentado.
- `acción` : debe comenzar en la misma línea que el patrón al que está relacionada y refiere a código a ejecutarse cuando se satisfaga el patrón en la entrada.

Sección de código de usuario

Esta sección opcional es copiada de forma literal al archivo `lex.yy.c` y es usada para rutinas de acompañamiento que llaman o son llamadas por el escáner.

Tokens Utilizados

Los siguientes tokens seran el resultado del análisis en la siguiente etapa.

Token	Descripción
`A_TITULO`	Tag de apertura de `titulo` (`<title>`).
`C_TITULO`	Tag de clausura de `titulo` (`</title>`).
`A_DESC`	Tag de apertura de `descripción` (`<description>`).
`C_DESC`	Tag de clausura de `descripción` (`</description>`).
`A_CAT`	Tag de apertura de `categoría` (`<category>`).
`C_CAT`	Tag de clausura de `categoría` (`</category>`).
`A_LINK`	Tag de apertura de `link` (`<link>`).
`C_LINK`	Tag de clausura de `link` (`</link>`).
`A_URL`	Tag de apertura de `url` (`<url>`).
`C_URL`	Tag de clausura de `url` (`</url>`).
`A_DER`	Tag de apertura de `derechos de autor` (`<copyright>`).
`C_DER`	Tag de clausura de `derechos de autor` (`</copyright>`).
`A_ALT`	Tag de apertura de `alto de imagen` (`<height>`).
`C_ALT`	Tag de clausura de `alto de imagen` (`</height>`).
`A_ANCHO`	Tag de apertura de `ancho de imagen` (`<width>`).
`C_ANCHO`	Tag de clausura de `ancho de imagen` (`</width>`).
`A_CANAL`	Tag de apertura de `canal` (`<channel>`).
`C_CANAL`	Tag de clausura de `canal` (`</channel>`).
`A_ITEM`	Tag de apertura de `item` (`<item>`).
`C_ITEM`	Tag de clausura de `item` (`</item>`).
`A_IMG`	Tag de apertura de `imagen` (`<image>`).
`C_IMG`	Tag de clausura de `imagen` (`</image>`).
`D_RSS`	Definición completa de RSS incluyendo versíon (`<rss version="2.0">`)
`C_RSS`	Tag de clausura (`</rss>`).
`D_XML`	Definición completa de XML incluyendo versión y codificación opcional (`<?xml version="2.0" encoding="UTF-8"?>`).
`ENLACE`	Coincidencia de la expresión regular encargada de detectar enlaces válidos.
`NUM`	Números naturales de cualquier longitud.
`CAD`	Caracteres alfabéticos sensibles a mayúsculas y acentos.

Una vez detectado el token se devuelve junto con la porción de la entrada que corresponde con el token almacenada en la variable puntero `yytext`.

```
    Token: Texto de entrada
```

Ej:

```
    -Enlace: https://www.google.com
```

En el caso de algunos tokens se añadirá si corresponde a una apertura o una clausura de sentencia.

Ej:

```
    -Apertura de item: <item>
    -Clausura de item: </item>
```

Parser

El archivo `.y` usado para la generación del código fuente del parser tiene 4 secciones:

```
%{
    Prólogo
%}

Declaraciones de Bison

%%
    Reglas gramaticales
%%

Epílogo
```

Prólogo

En esta sección se definen tipos semánticos y variables usadas en las acciones. También nos permite usar comandos de preprocesador y definición de macros.

Declaraciones de Bison

Aquí se deben declarar los terminales que el analizador sintáctico debe esperar del lexer y que serán usados para la construcción de los no terminales y la conformación de las producciones gramaticales.

También es posible definir el tipo semántico de los símbolos.

Reglas gramaticales

Las reglas usadas para construir los símbolos no terminales y que serán usadas para evaluar la entrada.

Epílogo

En esta sección se encuentra el código de usuario como definiciones de funciones declaradas en el prólogo.

Funciones auxiliares

- `eval_parse()`: toma la salida de la ejecución de `yyparse()` almacenada en la variable `salida` e imprime el mensaje correspondiente al resultado.

Obtención del intérprete

- `lex.l` : archivo que contiene las expresiones regulares que definen los tokens y las reglas a ejecutar para cada uno.
- `lex.yy.c` : archivo de código fuente del analizar léxico.
- `parser.y` : archivo donde se define la gramática y las funciones de control del analizador sintáctico.
- `parser.tab.c` : archivo de código fuente del analizador sintáctico.
- `parser.tab.h` : archivo donde se enumeran los tokens junto con su código que será usado por el parser para reconocerlos.

Ejecución del intérprete

Ejemplos

Ejemplo nro. 1:

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
    <channel>
        <title>RSS de la catedra de Sintaxis y Semantica de Lenguajes</title>
        <link>https://frre.cvg.utn.edu.ar/course/view.php?id=399</link>
        <description>Sintaxis y Semantica de Lenguajes de la U.T.N.F.R.Resistencia.
</description>
        <item>
            <title>Planificacion 2022</title>
            <link>https://google.com</link>
            <description>Planificacion de catedra, con cronograma de clases y
evaluaciones</description>
        </item>
        <item>
            <title>Guia de Trabajos practicos</title>
            <link>https://frre.cvg.utn.ar</link>
            <description>Guia de ejercicios propuestos a resolver en clase
practica</description>
        </item>
            <item>
            <title>Enunciado TPI</title>
            <link>https://wl.edu</link>
            <description>Trabajo practico integrador</description>
        </item>
    </channel>
</rss>
```

Ejemplo nro. 2:

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
    <channel>
        <title> RSS de la cátedra de Sintaxis y Semántica de Lenguajes </title>
        <link>https://frre.cvg.utn.edu.ar/course/view.php?id=399</link>
        <description>Sintaxis y Semántica de Lenguajes de la U.T.N. F.R.Resistencia.
</description>
        <category>Educacion</category>
        <copyright>2022 UTN. FRRe. Licencia Creative Commons. Atribución-No Comercial-Compartir
Igual(CCBY-NC-SA)</copyright>
        <image>

<url>https://frre.cvg.utn.edu.ar/pluginfile.php/29750/theme_snap/coverimage/158439147course-
image.gif</url>
            <title>encabezado imagen SSL</title>
            <link>https://frre.cvg.utn.edu.ar/course/view.php?id=399</link>
            <height>250</height>
            <width>120</width>
        </image>
        <item>
            <title>Planificacion 2022</title>
            <link>https://google.com</link>
            <description>Planificacion de catedra, con cronograma de clases y
evaluaciones</description>
            <category>Planificacion</category>
        </item>
        <item>
            <title>Guia de Trabajos practicos</title>
            <link>https://frre.cvg.utn.edu.ar/mod/resource/view.php?id=43544&redirect=1</link>
            <description>Guía de ejercicios propuestos a resolver en clase
practica</description>
            <category>Practica</category>
        </item>
        <item>
            <title>Enunciado TPI</title>
            <link>https://wl.ar</link>
            <description>Enunciado del Trabajo práctico integrador</description>
            <category>Teoria</category>
        </item>
    </channel>
</rss>
```

Contraejemplo 1:

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
    <channel>
        <title>RSS de la catedra de Sintaxis y Semantica de Lenguajes</title>
        <link>https://frre.cvg.utn.edu.ar/course/view.php?id=399</link>
        <description>Sintaxis y Semantica de Lenguajes de la U.T.N.F.R.Resistencia.
</description>
        <item>
        </item>
    </channel>
</rss>
```

Conclusiones

Este trabajo nos ha ayudado a desarrollar un mejor entendimiento sobre aceptores en la práctica como también sobre las consideraciones que se deben tomar en cuenta a la hora de implementarlos en un escenario real con usos reales.

Nos brindó la oportunidad de desarrollar nuestros conocimientos en el lenguaje `C` y la forma en que un proyecto debe ser estructurado al mismo tiempo que se deben considerar múltiples partes individuales y sus interacciones.

Referencias

- Manual de bison.
- Manual de Flex.
- Flex & Bison - John Levine.
- Tester de expresiones regulares.