CXML: Intérprete para XML

Concha Medina Edgard José, Del Corro Gonzalo, Leiva Mario
Dpto. de Informática. Facultad de Ciencias Exactas y Tecnologías, Universidad Nacional de Santiago del Estero.
{edgard.007.85, delcorrog, mario.leiva.al}@gmail.com

RESUMEN

El lenguaje XML, de creación relativamente reciente, está siendo muy utilizado en aplicaciones web y en las comunicaciones entre servidores web, y adquiriere importancia en el área de las bases de datos, no sólo como soporte para la transferencia de datos sino como formato de almacenamiento.
Ya que el lenguaje XML es un estándar internacional y dado su importancia, en el año académico 2013 se solicitó a los alumnos de la asignatura Lenguajes de Programación y Compiladores de la Licenciatura en Sistemas de Información que se dicta en la Universidad Nacional de Santiago del Estero, el diseño y desarrollo de un intérprete que sea capaz de reconocer archivos XML bien formados. Las principales funciones que se implementaron son: lectura del texto con las etiquetas en el lenguaje de marcado, comprobación sintáctica del texto, emisión de los mensajes de error correspondiente y visualización por pantalla (informe) de los datos ingresados en el archivo XML.
Para el desarrollo del intérprete, se utilizaron generadores automáticos para el análisis léxico y sintáctico: FLEX y BISON, la programación de las operaciones especificadas se realizó en el lenguaje C y el desarrollo de la interfaz gráfica se llevó a cabo con Java.

Palabras Claves: intérprete, lenguaje XML, analizador léxico, analizador sintáctico

1. INTRODUCCIÓN

XML *(Extensible Markup Language)* es un lenguaje de marcado de carácter general. Creado en 1998 por W3C *(World Wide Web Consortium)*, es un estándar internacional libre y gratuito. Es un lenguaje informático que utiliza marcas o etiquetas para definir la estructura, presentación y/o formato de los textos. La palabra *marcado* obedece a que el lenguaje permite añadir etiquetas al contenido original del texto, y éstas poseen una función determinada: la de permitir que los programas informáticos puedan procesar o interpretar adecuadamente los textos [10].
El objetivo fundamental de XML es intercambiar datos estructurados entre sistemas de información, fundamentalmente a través de Internet. Se trata de un formato de texto plano, lo que facilita la transferencia de información, y asegura la independencia con respecto a las diferentes plataformas.
Para que los documentos XML sean procesables deben estar *bien formados* lo que implica que deben cumplir estrictas normas sintácticas. El modelo de datos de los documentos XML es jerárquico y está formado por dos estructuras principales: elementos y atributos. Los elementos pueden ser simples o complejos. Los elementos simples están formados por valores de datos básicos (cadenas de caracteres o strings); los complejos, por el contrario, suelen contar con una estructura

jerárquica formada por otros elementos. Los atributos son utilizados para describir información complementaria (metainformación).
La importancia de XML es alta en la tecnología web actual, ya que es la base de numerosos procesos y técnicas. XML se utiliza para marcar documentos de carácter variado: bibliotecas digitales, corpus textuales, en la representación y transferencia de información del comercio electrónico, entre otros usos. Asimismo, su uso en las bases de datos se ha incrementado notoriamente, no sólo como soporte para la transferencia de datos sino como formato de almacenamiento.
Por otra parte, en la asignatura Lenguajes de Programación y Compiladores de la carrera de Licenciatura en Sistemas de Información de la Universidad Nacional de Santiago del Estero, todos los años se solicita el diseño y construcción de un intérprete aplicado a distintas temáticas. En el año académico 2013, la cátedra solicitó como actividad de resolución de problemas del mundo real, el diseño y desarrollo de un intérprete que reconozca archivos XML bien formados, mediante la lectura de textos con formato XML y la devolución por pantalla, de un informe con los datos registrados en el texto XML. Se utilizaron generadores automáticos para el análisis léxico y sintáctico y la programación para las operaciones especificadas se realizaron en el lenguaje C.
Existen numerosos desarrollos de intérpretes en distintas áreas de aplicación, especialmente se cita a [9] que construye un intérprete para el lenguaje XML.
Este trabajo, por lo tanto, transmite una experiencia desarrollada como parte de la formación académica práctica en el área de interés, y se estructura de la siguiente manera: el Lenguaje XML, sus características y estructura se presenta en la Sección 2. En la Sección 3 se describe brevemente la metodología empleada. En la Sección 4 se presenta sintéticamente el desarrollo de los analizadores léxico y sintáctico, en la sección 5 se exponen las conclusiones y trabajos futuros, mientras que en la Sección 6 se consigna la bibliografía utilizada.

2. LENGUAJE XML. CARACTERÍSTICAS Y ESTRUCTURA

XML [8] es un lenguaje utilizado para almacenar datos en forma legible. Deriva del lenguaje SGML y permite definir la gramática de lenguajes específicos para estructurar documentos grandes. A diferencia de otros lenguajes, XML da soporte, entre otras, a bases de datos, hojas de cálculo, y es útil cuando varias aplicaciones se deben comunicar entre sí o integrar información.
Un documento XML está formado por el prólogo y por el cuerpo del documento así como texto de etiquetas [3,5].

- **Prólogo**: aunque no es obligatorio, los documentos XML pueden empezar con unas líneas que describen la versión XML, el tipo de documento, etc.
  El prólogo de un documento XML contiene:
  - ✓ Una declaración XML. Es la sentencia que declara al documento como un documento XML.
  - ✓ Una declaración de tipo de documento. Enlaza el documento con su DTD (definición de tipo de documento), o el DTD puede estar incluido en la propia declaración o ambas cosas al mismo tiempo.
  - ✓ Uno o más comentarios e instrucciones de procesamiento.
- **Cuerpo**: a diferencia del prólogo, el cuerpo no es opcional. Debe contener sólo un

elemento raíz, característica indispensable también para que el documento esté bien formado.

- **Elementos**: los elementos XML pueden tener contenido (más elementos, caracteres o ambos), o bien ser elementos vacíos, y se los señala mediante etiquetas. Una etiqueta consiste en una marca hecha en el documento, que señala una porción de éste como un elemento. Un elemento es una pieza de información con un sentido claro y definido. Las etiquetas tienen la forma <nombre>, donde nombre es el nombre del elemento que se está señalando.
- **Atributos**: los elementos pueden tener atributos, que son una manera de incorporar características o propiedades a los elementos de un documento. Deben ir entre comillas.
- **Entidades predefinidas**: son entidades para representar caracteres especiales para que, de esta forma, no sean interpretadas como marcado en el procesador XML.
- **Secciones CDATA**: es una construcción en XML para especificar datos utilizando cualquier carácter sin que se interprete como marcado XML. Permite que caracteres especiales no rompan la estructura.
- **Comentarios**: tienen el siguiente formato: *<!--- Esto es un comentario --->*

3. METODOLOGÍA

Para el diseño y desarrollo del intérprete se siguió la siguiente metodología:

1. **Exploración bibliográfica** sobre los temas involucrados en este trabajo: lenguaje XML: su estructura y sintaxis, traductores, etapas de un compilador y generadores automáticos léxicos y sintácticos.
2. **Construcción del Intérprete**
   - **2.1 Análisis**
     - **2.1.1 Análisis léxico (AL)**: especificación del vocabulario, de los componentes léxicos (identificadores, palabras claves, símbolos especiales, operadores, etc.), de las expresiones regulares para cada componente léxico. Codificación mediante el empleo del generador automático de analizadores léxicos: Flex. Prueba del analizador léxico.
     - **2.1.2 Análisis sintáctico (AS):** elaboración de la gramática y prueba manual de la misma, codificación mediante el empleo del generador de analizador sintáctico Bison. Prueba del analizador sintáctico.
   - **2.2 Síntesis**
     A partir de la gramática generada para las funciones, se realiza la implementación del intérprete en el lenguaje C.
3. **Evaluación del Intérprete**
   3.1 Pruebas unitarias para todas las operaciones detalladas.
   3.2 Pruebas de integración.

4. DESARROLLO DE CXML

4.1 ANÁLISIS

El intérprete debe realizar una serie de operaciones que se describen a continuación.

- Ingreso del texto con las etiquetas en el lenguaje de marcado.
- Verificación para comprobar si el texto está sintácticamente correcto. Un

documento se denominada «bien formado» si cumple con todas las definiciones básicas de formato y puede, por lo tanto, analizarse correctamente por cualquier AS. Los documentos deben:

- ✓ Tener una estructura estrictamente jerárquica con lo que respecta a las etiquetas que delimitan sus elementos. Una etiqueta debe estar correctamente incluida en otra, es decir, las etiquetas deben estar correctamente anidadas. Los elementos con contenido deben estar correctamente cerrados.
- ✓ Los documentos XML sólo permiten un elemento raíz.
- ✓ Los valores atributos en XML siempre deben estar encerrados entre comillas simples o dobles.
- ✓ XML es sensible a mayúsculas y minúsculas. Existe un conjunto de caracteres llamados espacios en blanco (espacios, tabuladores, retornos de carro, saltos de línea) que los procesadores XML tratan de forma diferente en el marcado XML.
- ✓ Es necesario asignar nombres a las estructuras, tipos de elementos, entidades, elementos particulares, etc.
- ✓ Las construcciones como etiquetas, referencias de entidad y declaraciones se denominan marcas; son partes del documento que el procesador XML espera entender. El resto del documento entre marcas son los datos «entendibles» por las personas.

- Emisión de los mensajes de error correspondientes.
- Muestra por pantalla (en un informe) los datos ingresados en el archivo XML.

4.1.1 ANÁLISIS LÉXICO

4.1.1.1 Descripción de componentes léxicos

El analizador léxico genera unidades lógicas que se denominan componentes léxicos o tokens. El componente léxico es un par que consiste en un nombre de token y un valor de atributo, que puede ser opcional. En la mayoría de los lenguajes de programación se consideran componentes léxicos las siguientes construcciones: palabras claves, operadores, identificadores, constantes, cadenas literales y signos de puntuación. Existen componentes léxicos que representan un número finito de lexemas, entendiéndose como las cadenas de caracteres en el programa fuente que se pueden tratar como unidad léxica. Ejemplos de estos son: símbolos de puntuación, operados, palabras claves, etc. Sin embargo, un token puede representar un número infinito de lexemas, como por ejemplo los identificadores o constantes. Por lo tanto, el analizador léxico debe construir los valores para esos componentes léxicos y proporcionar información adicional sobre el lexema concreto. Esta información se denomina atributo del componente léxico**.** El conjunto de cadenas de la entrada se describe mediante una regla llamada patrón asociada al componente léxico. Para describir los patrones se utiliza la notación de expresiones regulares. Los lexemas para el token que concuerdan con el patrón representan cadenas de caracteres en el programa fuente que se pueden tratar como unidad léxica. Por lo tanto, un patrón es una regla que describe el conjunto de lexemas. [1, 2, 4]

En la tabla 1 se describen los componentes léxicos y las expresiones regulares para

cada uno de ellos que se desarrollaron para el lenguaje y que se utilizaron para la construcción del AL.

Tabla 1. **Descripción de los componentes léxicos y expresiones regulares de CXML**

COMPONENTE LÉXICO	DESCRIPCIÓN FORMAL (ER)	DESCRIPCIÓN INFORMAL DEL PATRÓN
TKN_INICIOPROLOGO	<?xml version	Secuencia de apertura del prólogo, que es opcional. Si se define el prólogo, se debe detallar la versión de XML. Puede ser 1.0 o 1.1.
TKN_FINPROLOGO	?>	Secuencia que indica el final del prólogo.
TKN_ETIQUETA	[a-zA-Z_][a-zA-Z0-9_\-\.]*	Las etiquetas contienen datos, en vez del formato del mismo. Todo documento XML está construido a partir de etiquetas, en donde existe una etiqueta de inicio y una etiqueta de fin, y datos entre las mismas.
TKN_ATRIBUTO	[a-zA-Z_][a-zA-Z0-9_\-\.]*	Un elemento puede tener un conjunto de atributos. Un atributo se define mediante el nombre del atributo, el signo "=" y el valor del atributo.
TKN_DATOS	[^\<\>\=\/\&\t]*	Los datos puede ser la información contenida entre las etiquetas.
TKN_ABRE	<	Símbolo de apertura de etiqueta
TKN_CIERRA	>	Símbolo de cierre de etiqueta
TKN_CIERRA2	/>	Símbolo para cerrar etiqueta en una línea
TKN_ABRE2	</	Símbolo de apertura para cerrar la definición de un elemento.
TKN_IGUAL	=	Operador relacional para asignar valores a los atributos.
TKN_HILERA	\"[^\"]*\"	Las hileras son los valores de los atributos. Están comprendidos entre comillas dobles (" ").
TKN_ENCODING	encoding	Opcional. Indica la codificación empleada en el lenguaje.
TKN_STANDALONE	standalone	Opcional. Indica si el archivo XML depende de la información alojada en otro archivo externo.
ESPACIOS	[\n\t]+	Espacios/tabuladores/saltos de líneas a consumir por el AL
COMENTARIOS	<!--"{DATOS}"-->	Comentarios a consumir por el AL

4.1.1.2 Implementación del Análisis léxico

El AL tiene como función principal leer carácter por carácter de la entrada y elaborar como salida una secuencia de componentes léxicos que utiliza el AS para hacer el análisis. Se puede construir un AL usando herramientas que lo generan automáticamente, como por ejemplos, Lex y Flex [7].

Para la construcción de este intérprete se utilizó Flex que es un generador de

analizadores léxicos que a partir de expresiones regulares definen el comportamiento del analizador y genera código en distintos lenguajes de programación, como C. El archivo de entrada a Flex se realizó en un archivo de texto con extensión "*.l", utilizando un bloc de notas. En la figura 1 se puede observar el archivo de entrada a Flex, con la definición de los patrones de los lexemas que se deben buscar a la entrada, y al lado de tales expresiones regulares, se detallan (en C) las acciones a ejecutar tras encontrar una cadena que se adapte al patrón indicado.

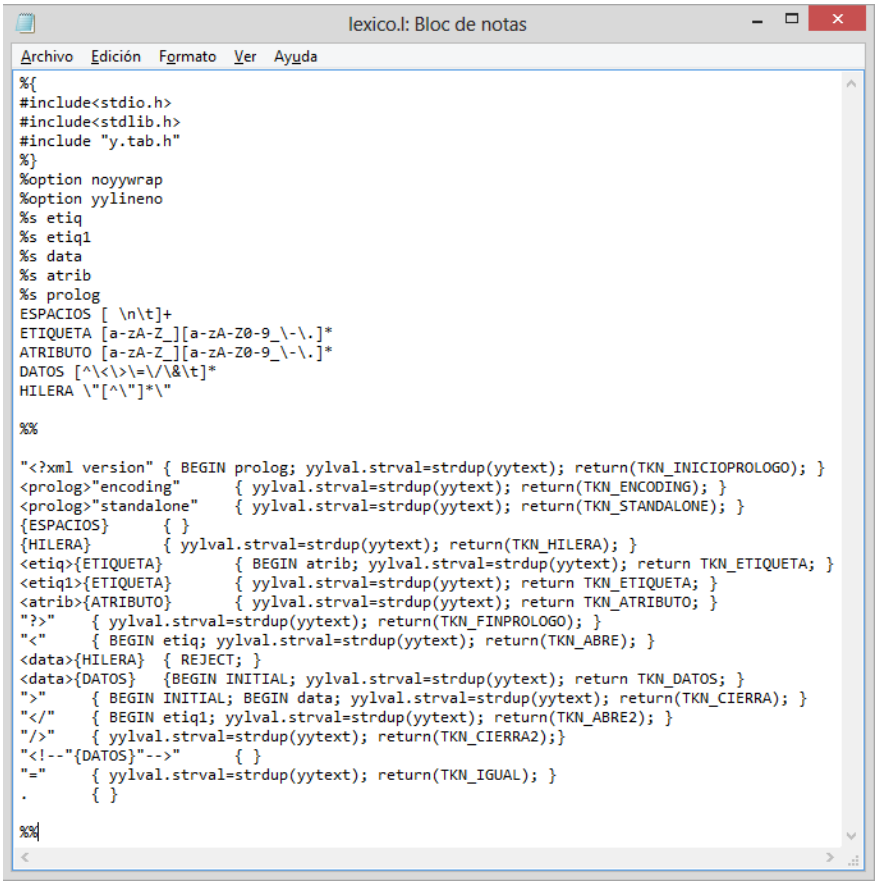

Figura 1: Archivo de entrada al Flex

4.1.2 ANÁLISIS SINTÁCTICO

4.1.2.1 Elaboración de la Gramática

Todo lenguaje de programación tiene reglas que prescriben la estructura sintáctica de programas bien formados y, dicha sintaxis se describe por medio de gramáticas libres de contexto. Para el intérprete CXML se define la siguiente gramática libre de contexto que responde a las especificaciones que debe cumplir un documento XML bien formado:

$G = (V_N, V_T, P, prologo)$ donde:

- El vocabulario de símbolos no terminal es: $\mathbf{V_N}$ = {prologo, versión, codificación, dependencia, raíz, expresión, data, atributo, exp2}
- El vocabulario de símbolos terminales está formado por: $\mathbf{V_T}$ = {TKN_INICIOPROLOGO, _FINPROLOGO, TKN_ETIQUETA, TKN_ATRIBUTO, TKN_DATOS, TKN_ABRE, TKN_CIERRA, TKN_CIERRA2, TKN_ABRE2, TKN_IGUAL, TKN_HILERA, TKN_ENCODING, TKN_STANDALONE}

- El conjunto de reglas de producción **P** se define como sigue:

prologo: **TKN_INICIOPROLOGO** version codificacion dependencia **TKN_FINPROLOGO** raiz
| raiz ;
version: **TKN_IGUAL TKN_HILERA**
codificación: **TKN_ENCODING TKN_IGUAL TKN_HILERA**
| λ **;**
dependencia: **TKN_STANDALONE TKN_IGUAL TKN_HILERA**
| λ**;**
raiz: **TKN_ABRE TKN_ETIQUETA** atributo **TKN_CIERRA** expresión data **TKN_ABRE2 TKN_ETIQUETA TKN_CIERRA**
expresion: **TKN_ABRE TKN_ETIQUETA** atributo **TKN_CIERRA** expresión data **TKN_ABRE2 TKN_ETIQUETA TKN_CIERRA**exp2
|**TKN_ABRE TKN_ETIQUETA** atributo **TKN_CIERRA2** expresion
| λ**;**
data: **TKN_DATOS**
| λ**;**
atributo: **TKN_ATRIBUTO TKN_IGUAL TKN_HILERA** atributo
| λ**;**
exp2: **TKN_ABRE TKN_ETIQUETA** atributo **TKN_CIERRA** expresion data **TKN_ABRE2 TKN_ETIQUETA TKN_CIERRA** exp2
|**TKN_ABRE TKN_ETIQUETA** atributo **TKN_CIERRA2** expresion
| λ;

La gramática ofrece las ventajas de definir en forma precisa el lenguaje y de permitir la construcción automática del AS.

4.1.2.2 Implementación del AS

El AS tiene como funciones principales [1, 2, 4]:

- Obtener una cadena de componentes léxicos del analizador léxico. y comprobar si la cadena puede ser generada por la gramática del lenguaje fuente.
- Informar los errores sintácticos en forma precisa y significativa. Deberá ser dotado de un mecanismo de recuperación de errores para continuar con el análisis.

Para la implementación del AS se utilizó el generador automático Bison que es un programa diseñado para compilar una gramática LALR(1) y producir el código fuente del AS del lenguaje generado por esta gramática. El archivo de entrada al Bison [6]

consta de un área de definiciones, un área de reglas que contiene la gramática y las acciones semánticas para las reglas y un área de funciones. Para la implementación del AS se realizó un archivo de texto con extensión "*.y", utilizando el bloc de notas.

4.2 SÍNTESIS

Los archivos XML se caracterizan por permitir almacenar información sin necesidad de seguir una estructura fija para todos los elementos. Las listas ofrecen la posibilidad de almacenar dicha información de forma dinámica, especialmente cuando no es posible saber de antemano cuántos elementos se almacenarán. Además, la utilización de una lista generalizada, permite controlar qué dato está relacionado con cada etiqueta, lo cual facilita generar el informe de salida.

Para llevar a cabo las operaciones, en el intérprete CXML se implementan: la lista "*etiquetas*", una lista generalizada para almacenar los nombres de las etiquetas o atributos y sus datos o valores asociados respectivamente. En la Figura 2 se representa un nodo de esta lista. El campo *etiqueta* contienen los nombres de las etiquetas o los valores de las mismas; *siguienteEt* y s*iguienteAtrib* son punteros hacia otros nodos. El puntero *siguienteEt*, solo es utilizado cuando el nodo contiene un nombre de alguna etiqueta o atributo.

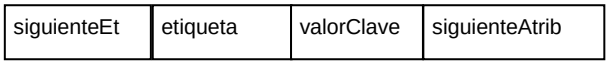

Figura 2: Definición del Nodo de la lista *Etiquetas*

Una vez de que se guarda en la lista el nombre de alguna etiqueta o atributo, si se encuentra un dato o un valor de atributo asociado a ese nombre, a través del puntero *siguienteAtrib* se direcciona hacia un nuevo nodo donde se guarda el dato. A medida que se encuentren más datos asociados a esa misma etiqueta se seguirán insertando en la lista a través del puntero siguienteAtrib.

Si se encuentra un nuevo nombre de etiqueta, a través del puntero *siguienteEt*, se direcciona a un nuevo nodo que contendrá el nombre de etiqueta encontrado. Estos pasos se repiten hasta que se acaba el archivo XML.

Al encontrar un atributo con valor asociado diferente al anterior, se cambia el valor de lo que se considera valor clave por el nuevo valor. En adelante todos los datos/valores encontrados se almacenan con este valor de clave. Al momento de encontrar un nuevo valor de clave, se almacena en la lista *listaClaves*. Con esta lista se puede conocer el número elementos definidos en el archivo XML. El nodo de la *listaClaves*, se muestra en la Figura 3.

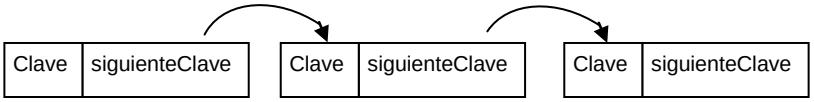

Figura 3: Definición de la lista Clave

4.3 PRUEBA

Para la prueba de CXML se escribieron varios documentos XML que fueron ingresados, a través de la interfaz gráfica (figura 4) para comprobar su correctitud. Se probaron tanto documentos bien formados como con errores.

En la figura 5 se puede observar un ejemplo de documento XML utilizado como entrada para la prueba y en la figura 6 se muestran la salida emitida por CXML.

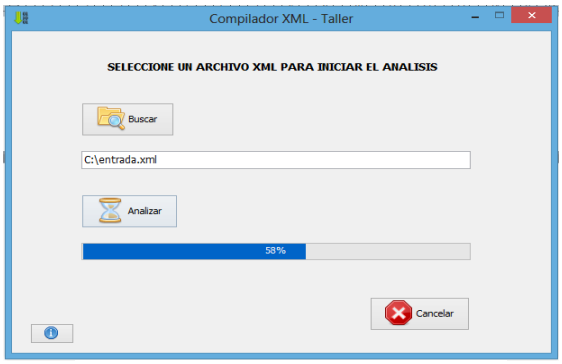

Figura 4: Interfaz Gráfica CXML

```
<?xml version="1.1"?>
<Libreria>
    <Libro ISBN = "555-666" Precio = "65" Edicion = "3era">
        <Titulo>Primer Curso BASE DE DATOS </Titulo>
        <Autores>
            <Autor>
                <Primer_nombre>Jeffrey </Primer_nombre>
                <Apellido>Ullman </Apellido>
            </Autor>
            <Autor>
                <Primer_nombre>jennifer </Primer_nombre>
                <Apellido> Widom </Apellido>
            </Autor>
            <Autor>
                <Primer_nombre>Edgar</Primer_nombre>
                <Apellido>Concha Medina</Apellido>
            </Autor>
        </Autores>
    </Libro>
</Libreria>
```

Figura 5: Entrada del CXML

```
Version XML ="1.1"
                    Parseo correcto: archivo sin errores

Cantidad de elementos: 13
Cantidad de atributos: 10
Etiqueta Clave: ISBN

ISBN              :"555-666"
Precio            :"65"
Edicion           :"3era"
Titulo           :Primer Curso BASE DE DATOS
Primer_nombre    :Jeffrey       jennifer     Edgar
Apellido          :Ullman       Widom        Concha Medina

*****************************************************************
```

Figura 6: Salida del CXML

5. CONCLUSIONES

En este trabajo se presentó el desarrollo de un intérprete denominado CXML que permite verificar si un documento XML está bien formado y emitir un informe de salida para dicho documento. La realización de este intérprete permitió, por un lado, profundizar los conceptos teóricos y prácticos adquiridos durante el año en las asignaturas Teoría de la Computación y Lenguaje de Programación y Compiladores. Por otro lado, adquirir habilidades y destrezas en el uso de los generadores automáticos como Flex y Bison y, en programación con lenguaje C. Pero además, se logró llevar los conceptos teóricos a la práctica computacional, para obtener como resultado una herramienta de software aplicable a problemas del mundo real.

Por último, ante el desafío presentado por la cátedra, se desarrollaron un conjunto de expresiones regulares, se codificaron en el fichero de entrada al Flex, se diseñó la gramática para el lenguaje XML que fue posteriormente codificada en Bison y, por último, se implementaron las rutinas y estructuras de datos necesarias para emitir el informe de salida. Es así que, se obtuvo un intérprete válido y completo, donde se verifica la sintaxis y se emite mensajes de errores si el archivo de entrada no es correcto.

Los trabajos futuros se orientan a realizar un editor de documentos XML que facilite las tareas de edición a los usuarios. Entre las funciones que se proponen es que posibilite la inserción de etiquetas de fin en forma automática luego de que el usuario haya ingresado una etiqueta de inicio. También se pretende generar un archivo con la estructura del documento XML ingresado, un archivo que contenga los elementos y atributos a definir, es decir, el esquema estructural del documento XML. Para esto se utilizará el lenguaje de definición de esquema DTD (Document Type Definition).

6. BIBLIOGRAFÍA

1. Aho, Sethi, Ullman. Principios, Compiladores: técnicas y herramientas. Addison Wesley, 2010.
2. Barchini de Gimenez, Graciela y Alvarez de Benitez Margarita. Fundamentos Teóricos de la Ciencia de la Computación, Departamento de Informática. FCEyT, 2009.
3. Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., & Yergeau, F. Extensible markup language (XML). World Wide Web Consortium Recommendation REC-xml-19980210. http://www.w3.org/TR/1998/REC-xml-19980210.
4. Louden, K. C. Construcción de compiladores: principios y práctica. Cengage Learning Editores, 2004.
5. Part, X. S.World Wide Web Consortium (W3C), Recommendation (October 28, 2004), http://www.w3.org/TR/xmlschema-1.
6. Pisabarro Marrón, Alma María."El generador de analizadores sintácticos YACC- Teoría de autómatas y lenguajes formales", Universidad de Valladolid
7. Simmross Wattenberg, Federico."El generador de analizadores léxicos lex – Teoría de Autómatas y Lenguajes formales", Universidad de Valladolid.

Otros Sitios de Consulta:

8. http://es.wikipedia.org/wiki/Extensible_Markup_Language
9. https://sites.google.com/site/emersalva/documentacion_a40235_tarea7#_Toc265082561
10. http://www.euskonews.com/0471zbk/gaia47101es.html