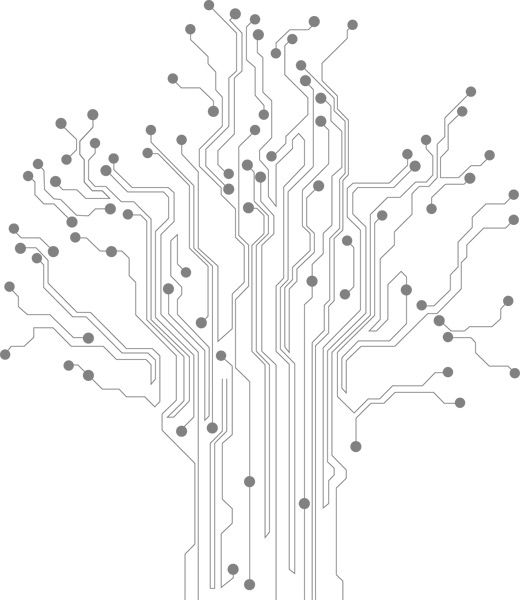
****

**Tecnológico de Costa Rica**

**Curso: IC-3101 Arquitectura de Computadores**

**Profesor: Erick Hernández Bonilla**

**Alumnos:**

**Adrián Garro Sánchez 2014088081**

**Liza Chaves Carranza 2013016573**

**Marisol González Coto 2014160604**

**Primer Proyecto: Validador de Documentos**

**Fecha de Entrega: 17 de abril**

**I Semestre 2016**

Tabla de Contenidos

[Introducción 3](#_Toc451115757)

[Arquitectura 4](#_Toc451115758)

[Algoritmos de Verificación de Consistencia 4](#_Toc451115759)

[Manual de Usuario 6](#_Toc451115760)

[Bibliografía 7](#_Toc451115761)

# Introducción

El desarrollo web ha tenido un apogeo constante en los últimos años, haciendo que nuevas tecnologías surjan para crear nuevas ideas y creativas páginas en el internet. Estas se basan en varios lenguajes, frameworks o formas para diseñar el contenido de las páginas, que utilizan lenguajes como CSS (Cascading Style Sheets), HTML (HyperText Markup Language) y XML (eXtensible Markup Language) siendo HTML el armazón, CSS el que le da los estilos, tipos de letras, colores, formas, y más y el XML el que transporta los datos de la página web.

La primera tarea asignada para el curso de Arquitectura de Computadores es crear un validador de documentos HTML y XML, que trabajan con *tags*. Estos objetos se abren y se cierran y tienen una estructura específica. Su posición, su apertura y cierre, su indentación y nombres es lo que se pretende validar con el programa a desarrollar.

# Arquitectura

El programa se va a desarrollar bajo la siguiente estructura:

# Algoritmos de Verificación de Consistencia

Apertura y Cerradura de Tags

El algoritmo para verificar la consistencia de los tags, está separado en varias secciones:

1. Verificar la apertura y cerradura de los tags:
   1. El primer paso consiste en verificar el caracter “<” que es el comienzo de un tag.
   2. Cuando se encuentra este, se sigue el proceso para buscar el cierre, el carácter “>”.
   3. Si se encuentra “<”, y encuentra “>”, proseguirá con los demás tags para encontrar el tag que cierra. Si no encuentra “>” para cerrar el tag, pero se encuentra otro “<” mostrará un error con la fila y columna correspondiente.
   4. Si no se encuentra “</”, mostrará un error en pantalla, con su fila y columna correspondiente. Si encuentra “</” pero no encuentra “>”, mostrará un error con la fila y la columna correspondientes.

Comillas

El algoritmo es similar al anterior, se revisa dónde comienza la comilla hasta la siguiente, solo que en esta fase, en las comillas existe una restricción de los caracteres que pueden ir dentro de ellas. Según la tabla ASCII, los caracteres que pueden ir dentro de las comillas van desde el 45, que corresponde al caracter “-” hasta el 122, que sería la zeta minúscula. Este proceso aplica tanto para comillas simples como para comillas dobles.

Tags Anidados

El algoritmo para la revisión de tags anidados se basa en una extensión del algoritmo para la verificación de apertura y cerradura de tags. Esta extensión consiste en copiar el contenido de un tag previamente verificado, seguidamente se busca en forma exhaustiva otro tag que posea el mismo contenido que el copiado originalmente, una vez que se comprueba esto se procede a verificar que este tag final posea el símbolo ‘/’ después su inicio (<). Si no se llega a encontrar este tag final, se informará en la consola este error con la línea y columna donde este se generó.

Identación de Líneas

TODO

Revisión de Comentarios

# Manual de Usuario

El programa se compila de la siguiente manera desde la línea de comandos:

*$>cd carpeta\_donde\_esta\_el\_programa*

*$>make*

Luego se ejecuta de esta manera:

* Para XML:

*$>documents\_analyzer < validate\_archive.xml*

* Para HTML:

*$>documents\_analyzer < validate\_archive.html*

Para cada uno de los casos, se mostrarán los errores que contenga el archivo ingresado, con su respetiva justificación, asímismo la línea y la columna donde se dieron.Si el archivo no posee errores, se mostrará en la pantalla el nuevo archivo con las identaciones necesarias, si aplica.

# Bibliografía

NASM. *NASM Manual.* Recuperado el 06 de mayo de 2016, desde <http://www.nasm.us/doc/>

Wikibooks. *X86 Assembly/Control Flow - Wikibooks, open books for an open world.* Recuperado el 07 de mayo de 2016, desde <http://en.wikibooks.org/wiki/X86_Assembly/Control_Flow>

Code as Art. *Code as Art: Assembly x86\_64 programming for Linux.* Recuperado el 08 de mayo de 2016, desde <http://0xax.blogspot.com/p/assembly-x8664-programming-for-linux.html>