Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Organización de Lenguajes y Compiladores 1
Primer Semestre 2019
Ing. Mario Bautista
Ing. Manuel Castillo
Ing. César Batz

Primer Proyecto

OBJETIVOS

General

Que el estudiante ponga en práctica los conocimientos aprendidos en clase y laboratorio para el desarrollo de las fases de análisis léxico y sintáctico de un compilador, utilizando herramientas generadoras de analizadores léxicos y sintácticos.

Específicos

- Que el estudiante aprenda a generar analizadores léxicos y sintácticos utilizando las herramientas [CUP y JFlex].
- Aplicar conocimientos sobre análisis léxico y sintáctico para la generación de soluciones de software.

DESCRIPCIÓN

Para este proyecto se requiere que el estudiante implemente un programa capaz de analizar y procesar un lenguaje de etiquetas con código embebido incrustado entre etiquetas y que genere una representación visual de todos los elementos que se definan en el programa de entrada en forma de página web HTML.

FUNCIONALIDADES

Se requiere un programa que permita la edición de programas fuente a través de un área de texto. Además, que permita la visualización de los resultados y reportes de análisis.

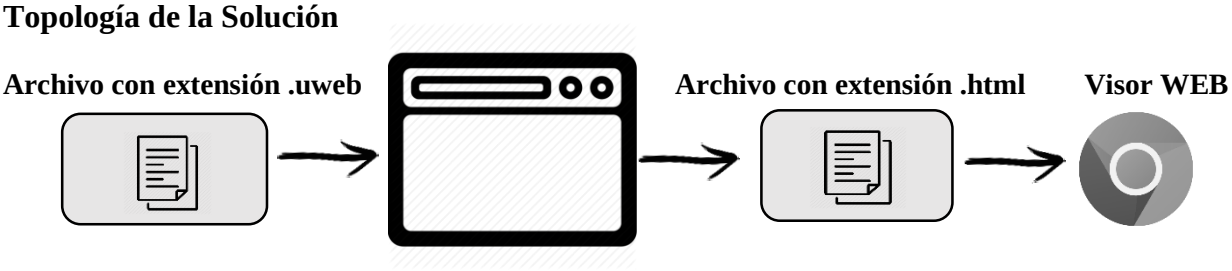
Menú Archivo

- Nuevo Archivo: Se podrá crear un nuevo archivo.
- Abrir archivo: Nos permitirá abrir un archivo que esté guardado en disco. Los archivos a abrir tendrán extensión [.uweb].
- Guardar: Nos permitirá guardar el archivo actual.
- Guardar como: Nos permitirá guardar el archivo actual con otro nombre.
- Compilar: Iniciará el análisis del archivo actual.

Menú Ayuda

- Manual de Usuario: Abrirá el manual de usuario del programa en formato pdf.
- Manual técnico: Abrirá el manual técnico del programa en formato pdf.
- Acerca de: mostrará los datos del estudiante.

Topología de la Solución



Área de Edición

Se requiere un cuadro de texto en donde se puedan visualizar y editar los archivos abiertos.

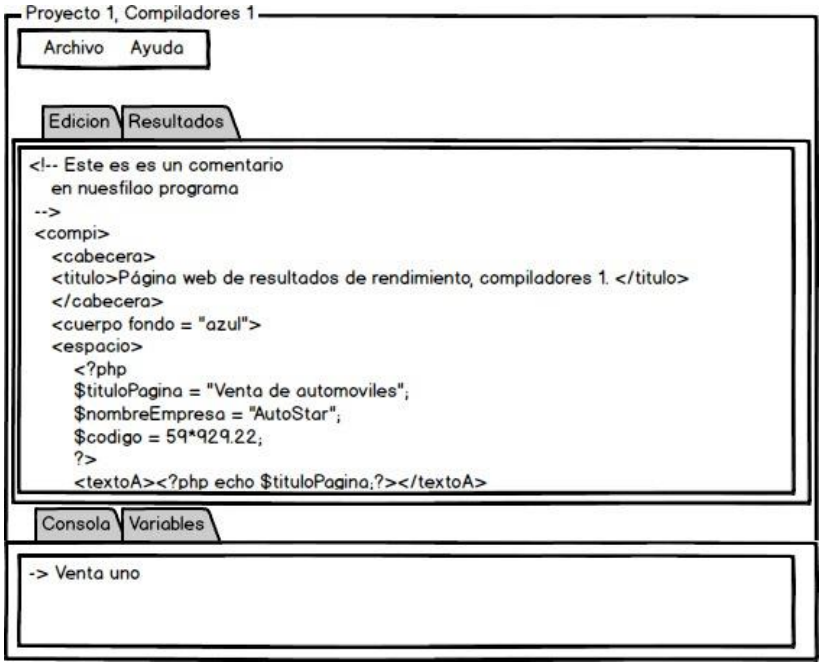


Figura No 1: Vista del área de edición de la aplicación.

Área de Resultados

Se requiere un área de resultados con la elección y la visualización de los siguientes reportes:

- Página Web Resultante: Mostrará la página web resultante.
- Análisis léxico: Mostrará la página web con el reporte de todos los tokens reconocidos durante el análisis léxico.
- Errores léxicos: Mostrará la página web con el reporte de todos los errores léxicos detectados durante el análisis.
- Errores Sintácticos: Mostrará la página web con el reporte de todos los errores sintácticos detectados durante el análisis.

Además, se contará con la opción de poder abrir la página HTML de cada uno de los resultados a través de un programa externo.

Área de salida

Se requiere de un área de salida en donde se encontrarán los siguientes elementos:

- Consola de salida: Es el área donde se mostrarán las impresiones de mensajes de consola indicadas en el programa fuente.
- Área de variables: Es el área donde se reportan todas las variables declaradas en el programa fuente, su tipo y su valor final.

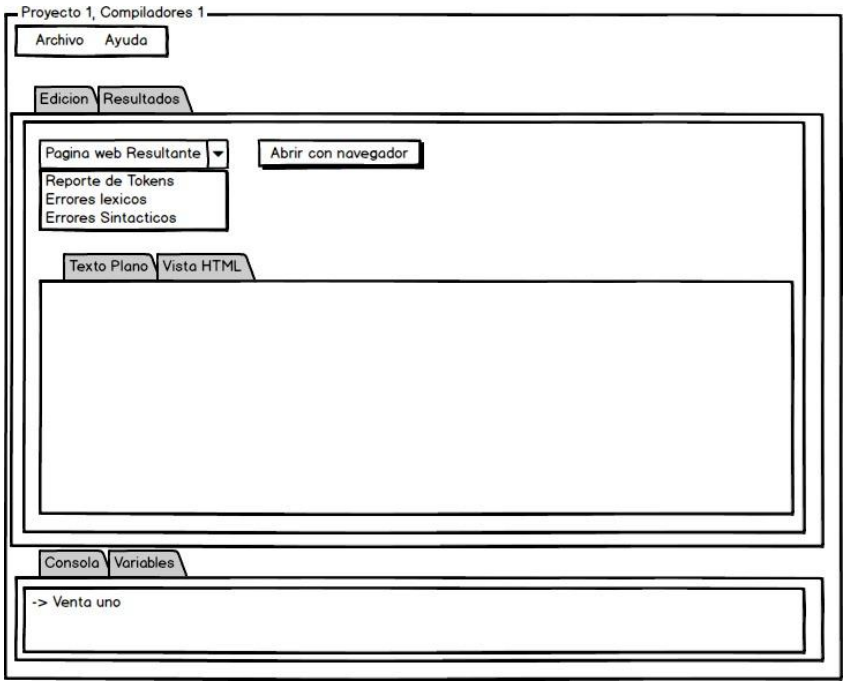


Figura No 2: Vista del área de resultados de la aplicación

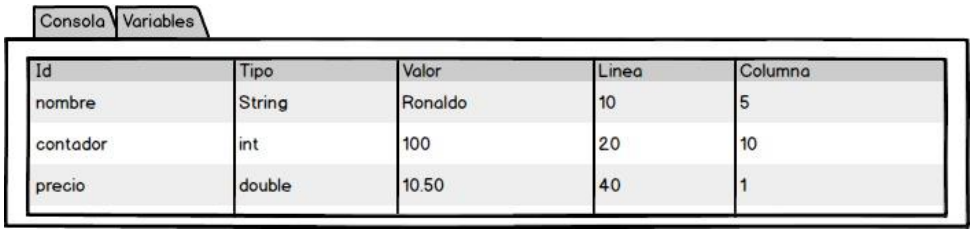


Figura No 3: Vista del área de salida de la aplicación

ESPECIFICACIÓN DEL PROGRAMA FUENTE

El programa fuente contará con dos lenguajes, Struct y HScript.
Struct será un lenguaje de etiquetas que permitirá la creación de la estructura de la interfaz resultante.
Pre será un lenguaje que permitirá la versatilidad de un lenguaje de programación tales como declarar variables y realizar operaciones con ellas.

Especificación del lenguaje Struct

Este lenguaje es un lenguaje de etiquetas, parecido a HTML y que se emplea para la construcción de páginas web. Cada instrucción tiene una etiqueta de apertura y una de cierre. Además, cada etiqueta contará con diferentes atributos. El lenguaje será ***case insensitive.*** es decir que no hace distinción entre mayúsculas y minúsculas.

Comentarios

El lenguaje Struct permitirá el uso de comentarios de una o varias líneas.
En ambos casos se utilizará la palabra <! para indicar el inicio del comentario y la palabra !> para indicar el final del comentario.

```
<! Este es un comentario
   en nuestro programa
!>
```

Etiquetas

Las etiquetas que componen el lenguaje Struct se mencionan a continuación:

<Compi> </Compi>

Esta etiqueta indica el inicio y el final del archivo. Dentro de esta etiqueta se encuentran las siguientes etiquetas:

- Cabecera
- Cuerpo

<Cabecera> </cabecera>

Indica el inicio y el final de la cabecera del archivo. Dentro de estas etiquetas se encuentra las etiquetas:

- Titulo

<Titulo> </Titulo>

Indica el inicio y el final del título de página resultante. El texto NO irá dentro de comillas dobles.
Ejemplo:

```
<COMPI>
        <CABECERA>
                <TITULO> Página web de resultados de rendimiento, compiladores 1
                </TITULO>
        </CABECERA>
        ….
</COMPI>
```

<Cuerpo> </Cuerpo>
Indica el inicio y el final del cuerpo de la página web resultante. Esta etiqueta también permite los siguientes atributos:

- Fondo: Este atributo indicará el color de fondo del cuerpo de la página web resultante. Su sintaxis es la siguiente:

```
fondo = "color"
```

  en donde "color" es el nombre propio de un color en idioma español o bien el color indicado en formato rgb.

Ejemplo:

```
<cuerpo fondo = "#f0f0f0"> (Contenido) </cuerpo>
<cuerpo fondo = "azul"> (Contenido) </cuerpo>
```

Además, dentro de esta etiqueta podrán venir las siguientes etiquetas:

- Párrafo
- Salto
- Tabla
- Imagen
- TextoA
- TextoB
- Botón
- Espacio
- **ScriptHS**

<espacio> </espacio>
Indica el inicio y el final de un espacio dentro de la página web resultante, se puede trabajar como una etiqueta de HTML. Dentro de esta etiqueta puede ir texto o bien las siguientes etiquetas:

- Párrafo
- Salto
- Tabla
- Imagen
- TextoA
- TextoB
- Botón
- Espacio
- **ScriptHS**

<parrafo> </parrafo>
Indica el inicio y el final de un párrafo. Esta etiqueta podrá contener los siguientes atributos:

- Alineación: Este atributo indicará la alineación del texto que contiene el párrafo. Los posibles valores de este atributo son los siguientes:
  - Izquierda
  - Derecha
  - Centrado
  - Justificado

Ejemplo:

```
<parrafo>
Este es un parrafo de prueba
</parrafo>
<parrafo alineacion = "justificado">

Classifying compilers by number of passes has its background in the hardware resource limitations of computers.
Compiling involves performing lots of work and early computers did not have enough memory to contain one program that did all of this work.
</parrafo>
```

</salto>
Indica un salto de línea. Puede estar dentro de la etiqueta cuerpo, párrafo, columna y columnaC.

<textoA></textoA>
Indica el inicio y el final de una cadena que se mostrará con un tamaño de fuente mayor al texto normal.

<textoB></textoB>
Indica el inicio y el final de una cadena que se mostrará con un tamaño de fuente mayor al texto normal. Pero menor al tamaño de la fuente del texto de la etiqueta textoA.

<imagen></imagen>
Indica que se mostrará una imagen en el lugar donde se coloque. Esta etiqueta tiene los siguientes atributos:

- Path [Obligatorio]: Indicará la ruta hacia la imagen a mostrar.
- Alto [Opcional]: Indicará la altura en píxeles de la imagen a mostrar.
- Ancho [Opcional]: Indicará el ancho en píxeles de la imagen a mostrar.

```
<imagen alto=30 ancho=30 path="imagen.jpg" > </imagen>
```

<boton> </boton>
Indica que se mostrará un botón en el lugar donde se coloque. Esta etiqueta tiene los siguientes atributos:

- Id [Obligatorio]: Será el identificador del botón.
- Texto [Obligatorio]: Indicará el texto a mostrar dentro del botón.

<tabla> </tabla>
Indica el inicio y el final de una tabla. Una tabla es un conjunto de filas, que a su vez son un conjunto de columnas. Esta etiqueta podrá contener los siguientes atributos:

- Borde: Aceptará dos posibles valores **true** o **false** el cuál indicará si la tabla tendrá un borde o no.

La etiqueta tabla podrá contener las siguientes etiquetas

- Fila

<Fila></fila>
Indica el inicio y el final de una fila dentro de una tabla. Esta etiqueta puede contener las siguientes etiquetas:

- columnaC
- columna

<columnaC> </columnaC>
Indica El inicio y el final de una columna, dentro de esta etiqueta puede venir texto libre o bien las siguientes etiquetas:

- Salto
- Párrafo
- Imagen
- Botón
- **ScriptHS**

Todo texto que venga dentro de esta etiqueta se mostrará en negrita.

<columna> </columna>
Indica el inicio y el final de una columna, dentro de esta etiqueta puede venir texto libre o bien las siguientes etiquetas:

- Salto
- Párrafo
- Imagen
- Botón
- **ScriptHS**

Ejemplo:

```
<tabla borde=true>
  <fila>
     <columnac>Codigo</columnac>
     <columnac>Tipo</columnac>
     <columnac>Modelo</columnac>
     <columnac>Marca</columnac>
     <columnac>Imagen</columnac>
  </fila>
  <fila> <columna>001</columna>
        <columna>Sedan</columna>
       <columna>2018</columna>
       <columna>Toyota</columna>
      <columna><imagen path="imagen.jpg" alto=30 ancho=30></columna>
  </fila>
  <fila> <columna>002</columna>
        <columna>Hatchback</columna>
        <columna>2030</columna>
        <columna>Ferrari</columna>
        <columna><imagen path="imagen.jpg" alto=30 ancho=30></columna>
   </fila>
</tabla>
```

ESPECIFICACIÓN DEL LENGUAJE HScript

El segundo lenguaje para analizar será un lenguaje de tipo script. Estos scripts pueden aparecer en las etiquetas mencionadas anteriormente. Para estos scripts se inicia con la palabra <?hs y se termina con la palabra ?>. Estos scripts pueden venir múltiples veces. Cada instrucción termina con punto y coma (;).

Case insensitive

El lenguaje HScript será un lenguaje case insensitive, es decir que no hace distinción entre mayúsculas y minúsculas.

Comentarios

El lenguaje HScript permitirá ingresar comentarios de una o de múltiples líneas. Para los comentarios de una línea se utilizará el símbolo // para indicar su inicio y su final será marcado por el salto de línea. Para el caso de los comentarios multilíneas se utilizará el símbolo /* para su inicio y */ para su final.

Ámbitos

El lenguaje HScript sólo podrá manejar un ámbito global.

Declaración de variables

Las variables se definen utilizando el símbolo de dólar ($) seguido de un identificador compuesto por letras, números o guiones bajos, no se define un tipo de dato y es obligatorio asignar un valor inicial, este valor definirá el tipo de dato de la variable. Los tipos que pueden asignarse son cadenas delimitadas por comillas dobles ("ejemplo"), enteros, decimales o booleanos (true, false).

Tipo	Ejemplo
Entero	30329
Decimal	99.99
Booleano	True False
Cadena	"Hola mundo"

Ejemplo:

```
<?hs
        $variable_entera = 5;
        $variable_decimal = 5.5;
        $variable_cadena = "ejemplo";
        $variable_booleana = true;
?>
```

Para las variables de tipo entero y decimal podrán asignarse operaciones aritméticas (+,-,*,/) y solo se realizarán con valores puntuales, no con otras variables, para el caso de enteros si el resultado de la operación es decimal deberá truncarse ese valor. En este proyecto no se validarán errores semánticos, por lo que no es necesario realizar las validaciones del caso.

Ejemplo:

```
<?hs
        $variable_decimal = 5*3/2+1-1;
        $variable_entero =9+9*5;
?>
```

Función echo

En el script de hs se utilizará la función nativa echo, que permitirá imprimir en consola cualquier variable o cadena, y para concatenar emplea el punto (.).

Ejemplo:

```
<cuerpo>
        <textoA>El siguiente código imprimirá variables de HR</textoA>
        <salto>
        <textoB>Hola
        <?HS   echo "Hola";
                echo $variable_entera;   // el valor de la variable es 7
        ?>
        </textoB>
        <?HS
                echo "numero ".$variable_entera;
        ?>
</cuerpo>
```

Que tendrá después de procesar esa parte de código, la salida siguiente:

Consola:

```
Hola
7
numero 7
```

SENTENCIAS DE CONTROL HScript

El lenguaje HScript soportará estructuras de control anidadas, las sentencias que se utilizarán en el proyecto son las siguientes:

If - Else

Se deberá implementar la sentencia de control if y else, donde la condición podrá contener operaciones de comparación (<, >, <=, >=, !=, = =) y lógicas (&&, ||, !), donde pueden usarse las variables definidas. Dentro de las instrucciones if y else no se realizará declaración de variables y se utilizará las definidas globalmente, puede aparecer cualquier instrucción y se debe manejar anidamiento. La sentencia else es opcional, puede o no aparecer. La estructura de la condición if es la siguiente:

Ejemplo:

```
<?hs
        if (5>1) {
                echo "es mayor";
                if (5>6){
                        echo "mucho mayor";
                }
        } else {
                echo "no es mayor";
        }
?>
```

Repetir

Se deberá implementar la sentencia de control repetir, a la que se le pasará como parámetro el número de veces que se debe realizar el código dentro de la sentencia, además pueden usarse las variables definidas. Dentro de las instrucciones repetir no se realizará declaración de variables y se utilizará las definidas globalmente, puede aparecer cualquier instrucción y se debe manejar anidamiento. La estructura de la condición repetir es la siguiente:

Ejemplo:

```
<?hs
        $var1 = 0;
        $var2 = 3;
        $var3 = 0;
        repetir (5) {
            echo "Repetir ".$var1;
            var1 = var1 + 1;
                repetir ($var2) {
                    echo "Repetir anidado".$var3;
                        var3 = var3 + 1;
                }
        }
?>
```

FUNCIONES NATIVAS DEL LENGUAJE Hscript

HScript permite la creación de variables Struct a través de las funciones que se describen a continuación utilizando la siguiente sintaxis:

```
#nombre_objeto = funcionNativa(argumentos);
```

Este nuevo tipo de variable tendrá los atributos descritos en el lenguaje Struct los cuales podrán ser establecidos a través de la función setValor(valor) y podrán ser obtenidos con la función getValor() de la siguiente manera:

Ejemplos:

```
#imagenPortada = CrearImagen("C://portada.jpg", 100, 10*10);
$path = #imagenPortada.getPath(); // Esto guardrá en la variable path el valor C://portada.jpg
$ancho = #imagenPortada.getAncho();
#imagenPortada.setPath("C://imagen/gato.jpg"); // Esto cambiará la ruta de la imagen
```

Función Insertar

La función imprimir indica que se insertará código de Struct donde se mande a llamar:

Ejemplo:

```
<?hs #imagenPortada = CrearImagen("C://portada.jpg", 100, 100); ?>
<?hs #imagenPortada.insertar();
#imagenPortada.insertar();   ?>
```

En el fragmento de código anterior se insertarán dos imágenes idénticas con los atributos de la variable imagenPortada.

Funciones para el párrafo:

Función CrearParrafo

La función CrearPárrafo creará una estructura párrafo al ser llamada dentro del lenguaje HScript. La función tendrá los siguientes parámetros, en orden:

- Contenido [String o Variable]: será el texto para mostrar dentro del párrafo
- Alineación [String o Variable, Opcional]: Este atributo indicará la alineación del texto que contiene el párrafo. Los posibles valores de este parámetro son los siguientes:
    - "Izquierda"
    - "Derecha"
    - "Centrado"
    - "Justificado"

También existirán las funciones **setContenido**(recibirá como parámetro un String o variable)**, setAlineación**(recibirá como parámetro un String o variable)**, getContenido**(retornará un String)**, getAlineación**(retornará un String)**, insertar.**

Ejemplo:

```
#parr1 = CrearParrafo("Este es un párrafo creado desde HScript", "justificado");
#parr2 = CrearParrafo("Soy otro párrafo");
#parr2.setAlineacion("centrado");
#parr2.setContenido($contenido);
$Var3 = parr2.getContenido();
$var4 = #parr1.getAlineacion();
#parr1.setAlineacion($var4);
#parr1.insertar();
```

Funciones para el textoA

Función CrearTextoA

Creará una cadena de texto igual a la que genera la etiqueta <textoA>, la función solo tendrá un parámetro que será texto o una variable y que representará el contenido de la etiqueta.

También existirán las funciones **setContenido**(recibirá como parámetro un String o variable)**, getContenido**(retornará un String), **Insertar.**

Ejemplo:

```
#title1 = CrearTextoA("Este es un texto de tamaño A");
$cad1 = #title1.getContenido();
#title1.setContenido("Otro contenido");
#title1.insertar();
```

Funciones para el textoB

Función CrearTextoB

Creará una cadena de texto igual a la que genera la etiqueta <textoB>, la función solo tendrá un parámetro que será texto o una variable y que representará el contenido de la etiqueta.

También existirán las funciones **setContenido**(recibirá como parámetro un String o variable) y **getContenido**(retornará un String)**, insertar.**

Ejemplo:

```
#subtitle1 = CrearTextoB("Este es un texto de tamaño B");
$cad1 = #subtitle1.getContenido();
#subtitle1.setContenido("Otro contenido");
#subtitle1.insertar();
```

Funciones para la imagen

Función CrearImagen

Creará una imagen en el lugar donde se coloque. Esta función tiene los siguientes parámetros, los cuales vendrán en el siguiente orden:

- Path [String o Variable, Obligatorio]: Indicará la ruta hacia la imagen a mostrar.
- Alto [Entero/Variable/Operación, Opcional]: Indicará la altura en píxeles de la imagen a mostrar.
- Ancho [Entero/Variable/Operación, Opcional]: Indicará el ancho en píxeles de la imagen a mostrar.

También existirán las funciones **setPath**(recibirá como parámetro un String o variable)**, getPath**(retornará un String)**, setAlto**(recibirá como parámetro un decimal, entero u operación), **getAlto**(retornará un decimal o entero), **setAncho**(recibirá como parámetro un decimal, entero, variable u operación), **getAncho**(retornará un decimal o entero), **insertar**.

Ejemplo:

```
#portada = CrearImagen("C://img.jpg", 34*2, 43(3));
#portada.setPath("C://img2.jpg");
$var3 = #portada.getAncho();
#portada.setAlto($var3);
#portada.setAncho(23.5+56-78*2);
#portada.insertar();
```

Funciones para la tabla

Función CrearTabla

Creará una tabla en el lugar donde se coloque. Esta función recibirá como parámetro una lista de filas las cuales a su vez tendrán una lista de elementos que se deberán mostrar en la tabla. Los elementos dentro de una fila pueden ser de cualquier tipo, por lo tanto se deberá hacer la conversión correspondiente a texto en caso de ser número o decimal. También existirán las funciones **setBorde**(recibirá como parámetro un Booleano o variable)**, insertar.**

Ejemplo de una fila:

```
["Elemento 1", $var_1, 45.3, 33]
```

Ejemplo de una tabla:

```
#tablaDatos = CrearTabla(
["Warner", "DC Comics", "Marvel", "LucasFilm", "Numeros"],
["Harry Potter", $var2, "Spiderman", "Episodio 4", 5.78],
["Inception", "Aquaman", "The Force Awakens", "EndGame", 8]
);
#tablaDatos.setBorde($booleanBorde);
#tablaDatos.insertar();
```

Warner	DC Comics	Marvel	LucasFilm	Numeros
Harry Potter	Justice League	Spiderman	Episodio 4	5.78
Inception	Aquaman	The Force Awakens	EndGame	8

Funciones para el botón

Función CrearBoton

Creará un botón en el lugar donde se coloque. Esta etiqueta tiene los siguientes parámetros:

- ID [String o Variable]: será el identificador del botón.
- Texto [String o Variable]: será el texto para mostrar dentro del botón.

También existirán las funciones **setTexto**(recibirá como parámetro un String o variable)**, getTexto**(retornará un String), **insertar.**

Ejemplo:

```
#boton1 = CrearBoton("IdBoton2", "Presióname");
#boton1.setTexto($var1);
$var2 = boton1.getTexto();
#boton1.insertar();
```

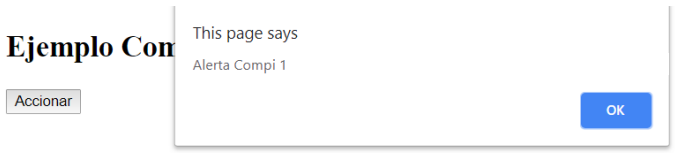
Función ClickBoton

La función ClickBotón creará una alerta de JavaScript en el archivo de HTML generado, esta función tendrá los siguientes parámetros:

- Texto [String o Variable]: será el texto para mostrar dentro de la alerta

Ejemplo:

```
# boton1.ClickBoton("Alerta Compi 1");
# boton2.ClickBoton("Alerta 2");
```



Expresiones lógicas, relacionales y aritméticas:

Estas expresiones se construyen a partir de operadores lógicos, relacionales y aritméticos, los operados admitidos son los siguientes:

OPERADORES LOGICOS	
Operador	**Descripción**
&&	AND, suma lógica
||	OR, multiplicación lógica
!	NOT, negación
OPERADORES RELACIONALES	
<	Menor que
>	Mayor que
<=	Menor o igual
>=	Mayor o igual
==	Igual
!=	Diferente
OPERADORES ARITMETICOS	
+	Suma
-	Resta
*	Multiplicación
/	División
SIGNOS DE AGRUPACION	
()	Paréntesis

Precedencia de análisis y operación de las expresiones lógicas y aritméticas:

Nivel	Operador
0	/ , *
1	+, -
2	==, !=, <, <=, >, >=
3	!
4	&&
5	||

Nota: 0 es el nivel de mayor importancia

Consideraciones

1. Trabajar de manera individual, utilizando Java con las herramientas JFlex y CUP.
2. El sistema operativo y el IDE son libres.
3. Cualquier copia total o parcial será reportada y no permitirá la continuidad en el laboratorio.
4. No se calificará desde el IDE, deberá generarse un ejecutable.
5. Los archivos de entrada serán proporcionados el día de la calificación
6. No habrá prórroga y entregas fuera de horario tienen una nota de CERO.
7. Fecha de entrega domingo 24 de marzo a las 23:59.

Entregables

Código fuente, ejecutable, manuales y archivos de prueba.