Manual de Usuario Generador de Captchas Ingeniería CUNOC (GCIC)

En el presente manual se pretende explicar al usuario final la utilización de la aplicación web Generador de Captchas Ingeniería Cunoc, o por sus siglas GCIC, que básicamente es una plataforma web que permite generar diferentes tipos de captchas a partir de un lenguaje de etiquetado parecido a HTML.

A continuación se explicará la estructura que deben de llevar las instrucciones para poder generar un captcha mediante GCIC. Las etiquetas GCIC, son las siguientes:

```
<C_GCIC>
<!-- Encabezado -->
  <C_HEAD>
    <C_TITLE>
      Titulo del Captcha
    </C_TITLE>

    <!-- Link al que redirige el captcha -->
    <C_LINK [href="https://github.com/cesar9401/CaptchaGenerator.git"]>
    </C_LINK>
  </C_HEAD>

  <!-- Cuerpo del captcha -->
  <C_BODY>
    <!-- Etiqueta scripting -->
    <C_SCRIPTING>
      <!-- proceso onload, se ejecuta cuando se lee el documento -->
      ON_LOAD () [
        integer @global contador = 0;
        ALERT_INFO("Intengo número: " + contador);
        contador = contador + 1;
      ]
    </C_SCRIPTING>

    <!-- Etiqueta de titulo, similar a <h1> -->
    <C_H1>Hola soy un titulo!</C_H1>

    <!-- Salto de linea -->
    <BR>

    <!-- Parrafo -->
    <C_P>Hola soy un parrafo</C_P>

    <!-- Similar a <span></span> de HTML -->
    <C_SPAM>Cuál es el resultado de 5 * 6 - 2</C_SPAM>

    <C_INPUT [type = "text"] [id="input_respuesta"]>

    <C_BUTTON [onclick="PROCESS_calcular()"] [id="button2"]>Enviar</C_BUTTON>


    <C_SCRIPTING>
      <!-- Proceso calcular se llama mediante el boton id="button2" -->
```

```
      PROCESS_calcular() [
        string resultado_input = getElementById('input_respuesta');
        string result = "28"

        IF(resultado_input == result) THEN
          INIT{:
            ALERT_INFO("El resultado es correcto");
            REDIRECT();
            EXIT();
          :}END
        ELSE
          INIT{:
            ALERT_INFO("Resultado incorrecto, intente otra vez");
          :}END

        INSERT('<C_P [id="parrafo1"] [color="navy"]>Soy un parrafo</C_P>');
      ]
    </C_SCRIPTING>
  </C_BODY>
</C_GCIC>
```

Como se ve en el ejemplo anterior, la estructura de un archivo .gcic es bastante simple, y similar a una estructura HTML. En el ejemplo anterior no se incluyeron las siguientes etiquetas, que según se pueden observar, tiene una similitud con etiquetas de HTML.

```
<!-- similar a <textarea></textarea> -->
<C_TEXTAREA [font-size = "12px"] [cols = "10"] [rows = "20"]></C_TEXTAREA>

<!-- Similar a select en HTML -->
<C_SELECT [id="soy un id"]>
  <C_OPTION>Opcion 1</C_OPTION>
  <C_OPTION>Opcion 2</C_OPTION>
  <C_OPTION>Opcion 3</C_OPTION>
  <C_OPTION>Opcion 4</C_OPTION>
</C_SELECT>

<!-- similar a etiqueta img en HTML -->
<C_IMG [srec="https://serielizados.com/wp-content/uploads/bojack-horseman-5-1-1024x576.jpg"]></C_IMG>

<!-- similar div en HTML -->
<C_DIV>
    <!-- Etiqueta de titulo, similar a <h1> -->
    <C_H1>Hola soy un titulo!</C_H1>

    <!-- Salto de linea -->
    <BR>

    <!-- Parrafo -->
    <C_P>Hola soy un parrafo</C_P>

    <!-- Similar a <span></span> de HTML -->
    <C_SPAM>Cuál es el resultado de 5 * 6 - 2</C_SPAM>

    <C_INPUT [type = "text"] [id="input_respuesta"]>
</C_DIV>
```

Como se ven en las descripciones anteriores, la estructura para un archivo .gcic es bastante simple y sigue las mismas especificaciones que las etiquetas HTML.

Más adelante se darán otros ejemplos, a continuación procederemos con algunos ejemplos de la estructura de los procesos entre las etiquetas `<C_SCRIPTING></C_SCRIPTING>`.

Tipos de variables y declaración de variables

Los tipos de variables existentes son: `integer`, `string`, `char`, `decimal` y `boolean` . Se tienen las siguientes observaciones para estos tipos de variables:

- Variables de tipo integer aceptan números positivos y negativos, 4 bytes maximo.
- Las variables de tipo decimal aceptan unicamente 4 cifras decimales
- Si se asigna un booleano a una variable de tipo integer, esta toma el valor de 1 en caso de true y 0 en caso de false.

Para declarar variables, se pueden seguir los siguientes ejemplos:

```
integer @global contador = 1;
decimal dec1, dec2 = 12.5, dec3;
decimal dec4;
string cadena = "hola soy una cadena";
char ch1 = 'a';
char ch2 = '?', ch3, ch4 = '!';
```

Modo @global

Es opcional y si se especifica en la declaración de una variable significa que esta variable no perderá su valor a cada llamada del procedimiento donde esté declarada, y su valor podrá ser actualizado cada vez que se reasigne en alguna parte del código a cada llamada del proceso o hasta que se cierre o se recargue de nuevo la página web que contiene el captcha

Operaciones aritméticas:

Se manejan las 4 principales operaciones aritméticas(suma, resta, multiplicación, división), siguiendo las siguientes pautas:

Suma:

Operación aritmética que consiste en reunir varias cantidades (sumandos) en una sola (la suma). El operador de la suma es el signo más (+).
Especificaciones:

- Al sumar dos datos numéricos (integer, decimal), el resultado será numérico

- Al sumar dos datos de tipo carácter (char, string), el resultado será la concatenación de ambos datos.
- Al sumar un dato numérico con un dato tipo char el resultado será la suma del dato numérico con la conversión ASCCI del dato de tipo char.
- Al sumar dos datos de tipo boolean, el resultado será un dato lógico, en este caso utilizaremos la suma como el análogo a la operación lógica OR.
- Todas las demás especificaciones se encuentran en la siguiente tabla.

+	integer	string	decimal	char	boolean
integer	integer	string	decimal	integer	integer
string	string	string	string	string	error
decimal	decimal	string	decimal	decimal	decimal
char	integer	string	decimal	integer	integer
boolean	integer	error	decimal	integer	boolean

Resta:

Operación aritmética que consiste en quitar una cantidad (sustraendo) de otra (minuendo), para averiguar la diferencia entre las dos. El operador de la resta es el signo menos (-). Especificaciones:

- Al restar dos datos numéricos (int, double) el resultado será numérico.
- En las operaciones entre número y carácter, se deberá convertir el carácter a código ASCII.
- No es posible restar datos numéricos con tipos de datos carácter (string).
- No es posible restar tipos de datos lógicos (boolean) entre sí.
- Todas las demás especificaciones se encuentran en la siguiente tabla.

-	integer	string	decimal	char	boolean
integer	integer	Error	decimal	integer	integer
string	Error	Error	Error	Error	Error
decimal	decimal	Error	decimal	decimal	decimal
char	integer	Error	decimal	integer	Error
boolean	integer	Error	decimal	Error	Error

Multiplicación:

Operación aritmética que consiste en sumar un número (multiplicando) tantas veces como indica otro número (multiplicador). El signo de Multiplicación será el asterisco (*). Las operaciones se podrán realizar sólo entre valores o variables del mismo tipo. Especificaciones:

Al multiplicar dos datos numéricos (int, double) el resultado será numérico.

- No es posible multiplicar datos numéricos con tipos de datos caracter (string)
- No es posible multiplicar tipos de datos string entre sí.
- Al multiplicar dos datos de tipo lógico (boolean), el resultado será un dato lógico, en este caso usaremos la multiplicación como operador AND entre ambos datos.
- Todas las demás especificaciones se encuentran en la siguiente tabla.

*	integer	string	decimal	char	boolean
integer	integer	Error	decimal	integer	integer
string	Error	Error	Error	Error	Error
decimal	decimal	Error	decimal	decimal	decimal
char	integer	Error	decimal	integer	integer
boolean	integer	Error	decimal	integer	boolean

División:

Operación aritmética que consiste en partir un todo en varias partes, al todo se le conoce como dividendo, al total de partes se le conoce como divisor y el resultado recibe él nombre de cociente. El operador de la división es la diagonal (/). Especificaciones:

- Al dividir dos datos numéricos (int, double) el resultado será numérico.
- No es posible dividir datos numéricos con tipos de datos caracter (string)
- No es posible dividir tipos de datos lógicos (boolean) entre sí.
- Al dividir un dato numérico entre 0 deberá arrojarse un error de ejecución.
- Todas las demás especificaciones se encuentran en la siguiente tabla.

/	integer	string	decimal	char	boolean
integer	decimal	Error	decimal	decimal	integer
string	Error	Error	Error	Error	Error
decimal	decimal	Error	decimal	decimal	decimal
char	decimal	Error	decimal	decimal	integer
boolean	decimal	Error	decimal	decimal	Error

Operadores Lógicos y Relacionales

Se manejan los operadores relacionales y lógicos siguientes:

OPERADOR	DESCRIPCIÓN
==	Es igual
!=	Es distinto
<, <=, >, >=	Menor, menor o igual, mayor, mayor o igual
&&	Operador and (y)
||	Operador or (o)
!	Operador not (no)

Para cada uno de los operadores aritméticos, lógicos y relacionales, se sigue la precedencia de signos que cualquier otro lenguaje de programación maneja, y el único símbolo que tiene una procedencia mayor son los paréntesis `()`.

Funciones Especiales

ASC

Recibe como parámetro una variable string, ordena la palabra en forma ascendente y la retorna como string.

```
string palabra = ASC("cadena");

string palabra2 = ASC("oracion 1" + palabra);
```

DESC

Recibe como parámetro una variable string, ordena la palabra en forma descendente y la retorna como string.

```
string palabra = DESC("cadena");

string palabra2 = DESC("oracion 1" + palabra);
```

LETPAR_NUM

Recibe como parámetro una variable string, codifica las letras pares de la palabra a su equivalente en ascci y la retorna como string.

```
integer id1 = 100;
string cadena = LETPAR_NUM("cadena 1" + id1);
```

LETIMPAR_NUM

Recibe como parámetro una variable string, codifica las letras impares de la palabra a su equivalente en ascci y la retorna como string.

```
string cadenita = LETIMPAR_NUM(ASC("sy una oracion ")) + "hola";
```

REVERSE

Recibe como parámetro una variable string, reescribe la palabra al revés y la retorna como string.

```
string rev = REVERSE(LETPAR_NUM(ASC("12345")));
```

CARACTER_ALEATORIO

No recibe parámetros y retorna un carácter aleatorio entre a y z o A y Z.

```
char ch1 = CARACTER_ALEATORIO();
```

NUM_ALEATORIO

No recibe parámetros y retorna un número entero entre 0 y 9.

```
integer entero1 = NUM_ALEATORIO() + NUM_ALEATORIO();
```

ALERT_INFO

Recibe como parámetro un mensaje tipo string, y despliega un mensaje emergente en la página web generada.

```
ALERT_INFO(REVERSE("12345"));
```

EXIT

Esta función lo que hace es parar la ejecución del scripting y automáticamente sale de la función donde se mande a llamar.

```
EXIT();
```

REDIRECT

Está función redirige a la página o link que se ha establecido en la etiqueta <C_LINK>, cuando es llamada.

```
REDIRECT();
```

getElementById

Para poder acceder a los identificadores de etiquetas y poder obtener por ejemplo el valor de un input, se cuenta con la función getElemenById('id_Elemento'), y la misma puede ser asignada a una variable tipo string únicamente.

```
string respuesta = getElementById('input_1');
```

INSERT

Con esta instrucción, se le indicará al intérprete que escriba al archivo de salida. La sintaxis es la siguiente:

```
INSERT(token1,token2...,tokenN);
```

También es posible insertar etiquetas:

```
INSERT('<C_INPUT [color="navy"] [id="input_1"] [type="text"]></C_INPUT>');

INSERT('<C_BR>');

INSERT('<C_BUTTON [id= "boton_1"] [onclick="PROCESS_calc1()"] [background="#F1C40F"]> Procesar calc1</C_BUTTON>');
```

Estructuras de control

IF

Esta instrucción recibe una condición booleana o una expresión que retorna un booleano y si esta se cumple se ejecutará la lista de instrucciones que viene a continuación de la sentencia. Así mismo pueden venir más sentencias ELSE IF después del primer IF y/o solamente la instrucción ELSE que no tiene ninguna condición a evaluar y es el comodín a la cual recurre la sentencia en el caso de que ninguna condición se cumpla.

IF con una sola instrucción:

```
IF(!var1 && var2 || 12 > 10) THEN
  integer entero1 = 12;
```

IF ELSE con varias instrucciones

```
IF(!var1 && var2 || 12 > 10) THEN
  INIT{:
    integer int1 = 12;
    integer int2 = 12;
    ALERT_INFO("El resultado es: " + int1 * int2);
  :}END
ELSE
  INIT{:
    integer numerito = true && true;
    INSERT("Condicion false");
  :}END
```

La estructura IF ELSE-IF ELSE, se trabaja de manera similar.

REPEAT

Este ciclo ejecutará el número de veces necesarias, hasta llegar a cierto límite especificado por una variable o una expresión numérica. Este tendrá un área de asignación o declaración, y otra área donde se especifique el límite.

```
REPEAT(integer i = 0) HUNTIL (12 + 12 - 3 * 4)
  INIT{:
    ALERT_INFO("Iteración: " + i);
    IF(i == 10) THEN
      EXIT();
  :}END
```

WHILE

Este ciclo ejecutará su contenido siempre que se cumpla la condición que se le dé por lo que podría no ejecutarse si la condición es falsa desde el inicio. La estructura del ciclo es la siguiente.

```
integer int1 = 0;
WHILE(int1 < 10) THENWHILE
  INIT{:
    INSERT('El valor es: ' + int1);
    int1 = int1 + 1;
  :}END
```

Comentarios

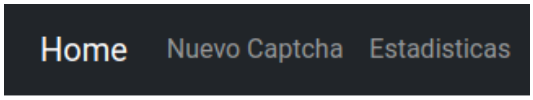
Se pueden usar los siguientes formatos para comentarios:

```
<!-- Hola soy un comentario -->

!! comentario de linea

<!--
    Comentario multilinea
    Hola soy otro comentario
-->
```

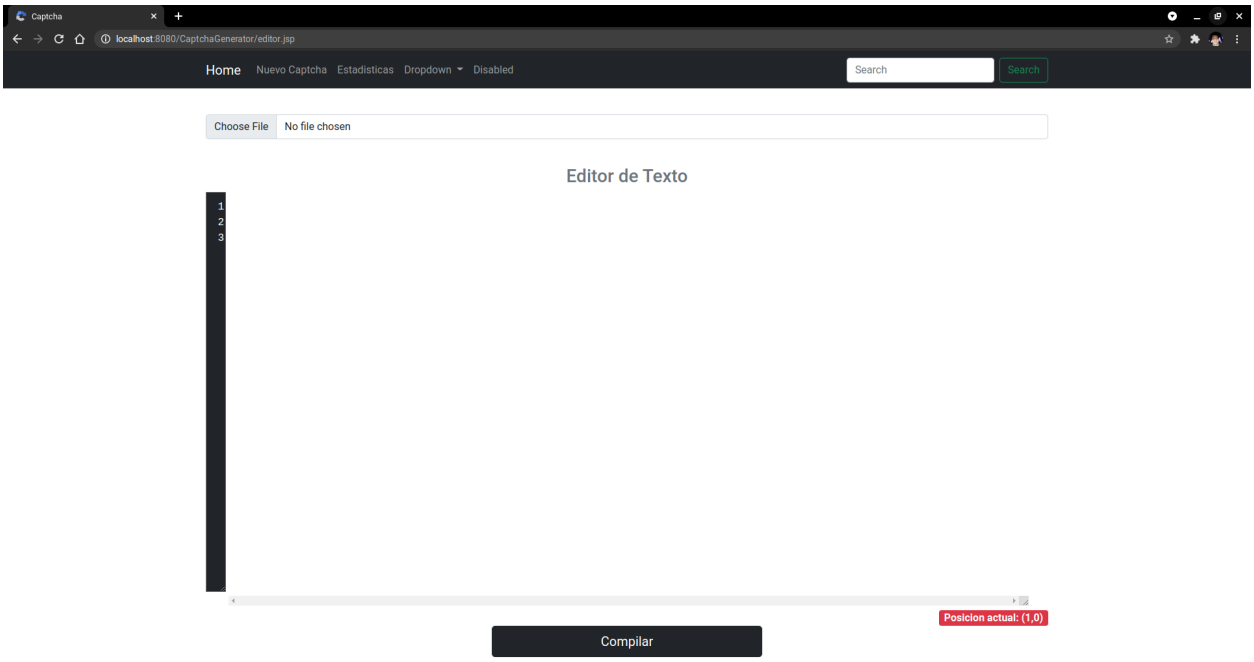
¿Cómo usar la aplicación web?

La vista principal es la siguiente:

La funcionalidad es bastante simple, para crear un captcha nuevo basta con tener ya listo nuestro archivo con extensión .gcic, aunque también se cuenta con un editor para craer nuestros archivos desde la aplicación, en ambos casos ubicamos el menu que dice nuevo captcha y hacemos click sobre el.

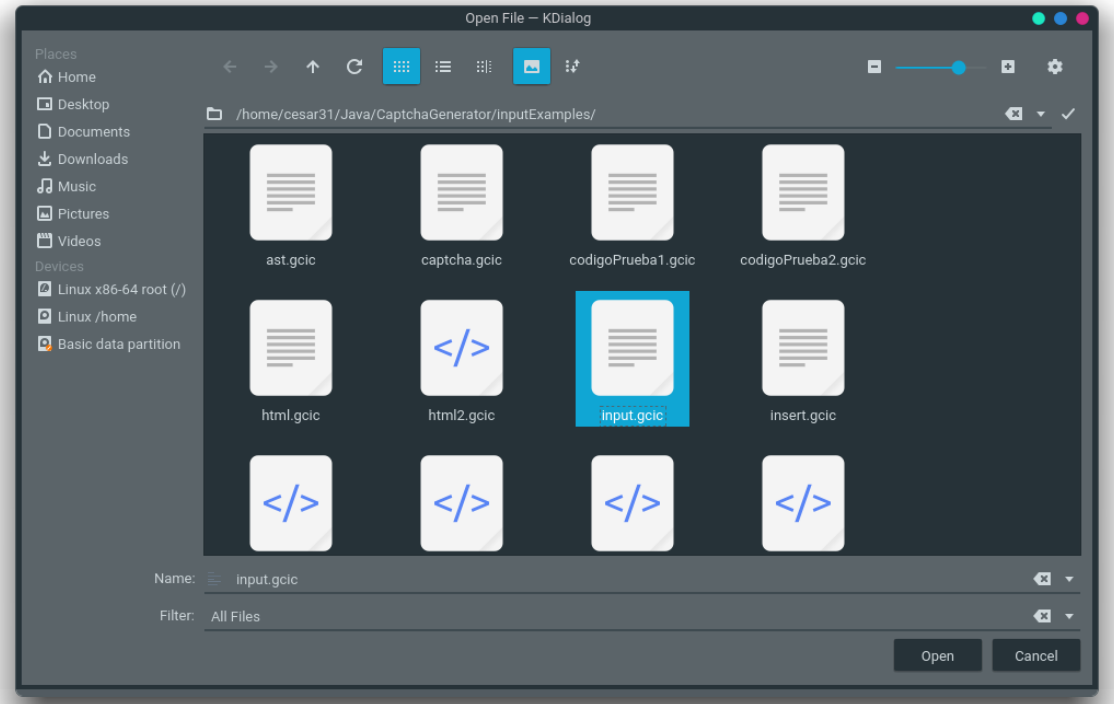


De inmediato vamos a la siguiente vista, donde contamos con un editor de texto, un input de tipo file para poder cargar nuestros archivos y un boton para poder ejecutar el proceso de compilación.

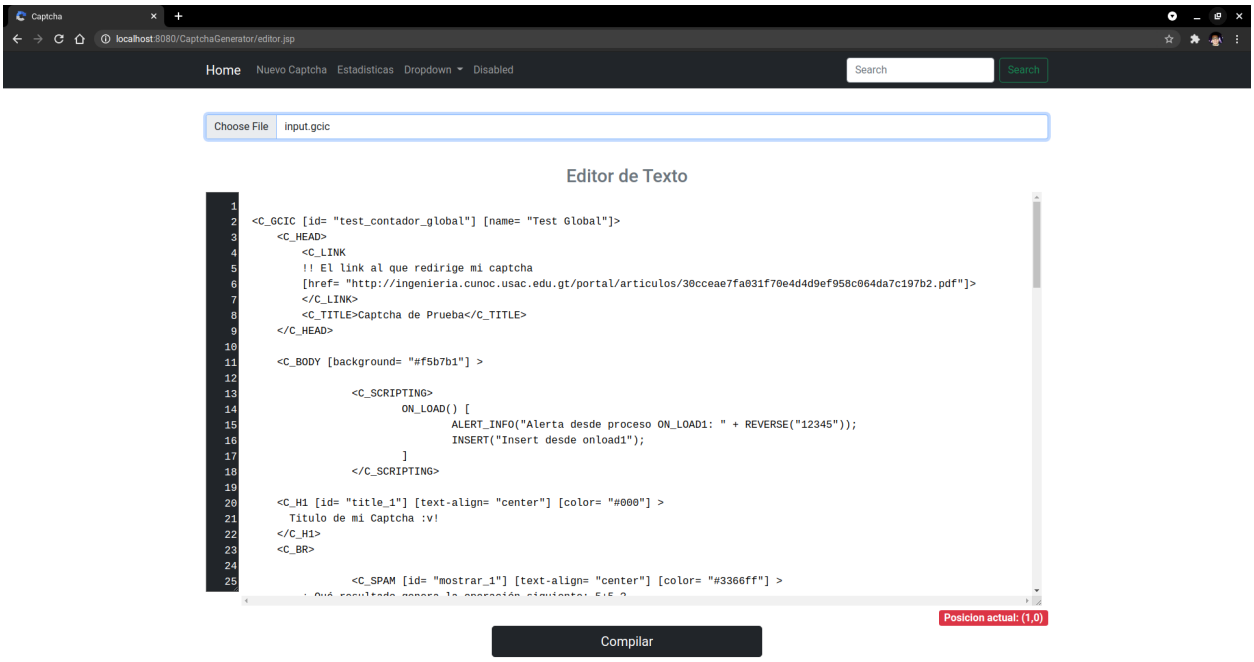


Para ejecutar un archivo que ya tenemos, basta con hacer click sobre el input de tipo file.

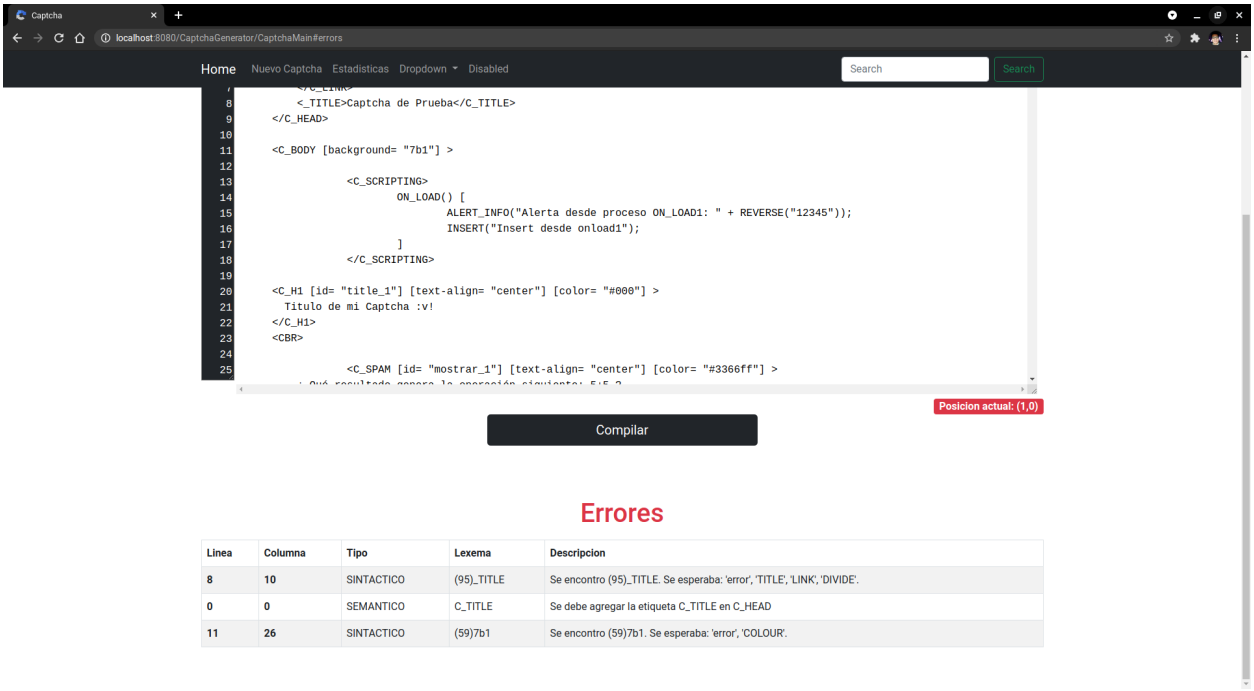
Se nos desplegará un buscador de archivos, seleccionamos el archivo que tenemos en nuestra computadora.



Seleccionamos nuestro archivo y damos clic en el botón abrir, a continuación visualizaremos las instrucciones en el editor de texto por si deseamos cambiar algo.

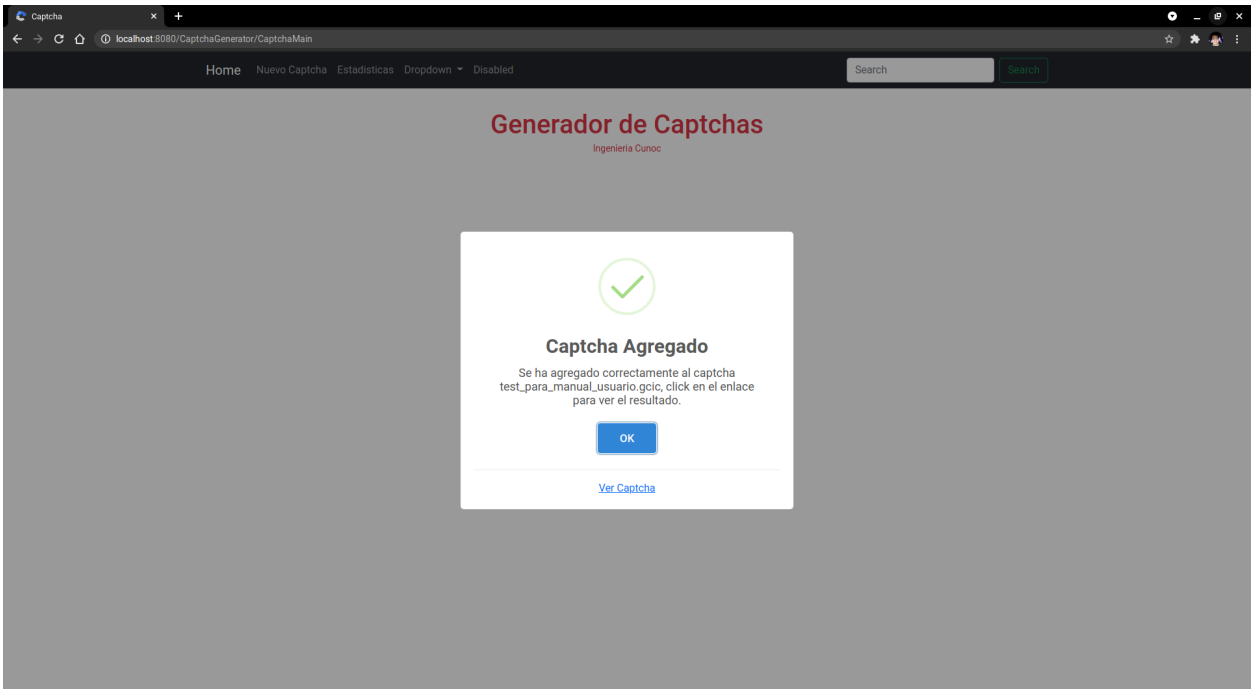


Si estamos seguros de la sintaxis y datos de archivo, procedemos a hacer clic en el botón compilar, en caso de errores, se nos desplegará la siguiente tabla.



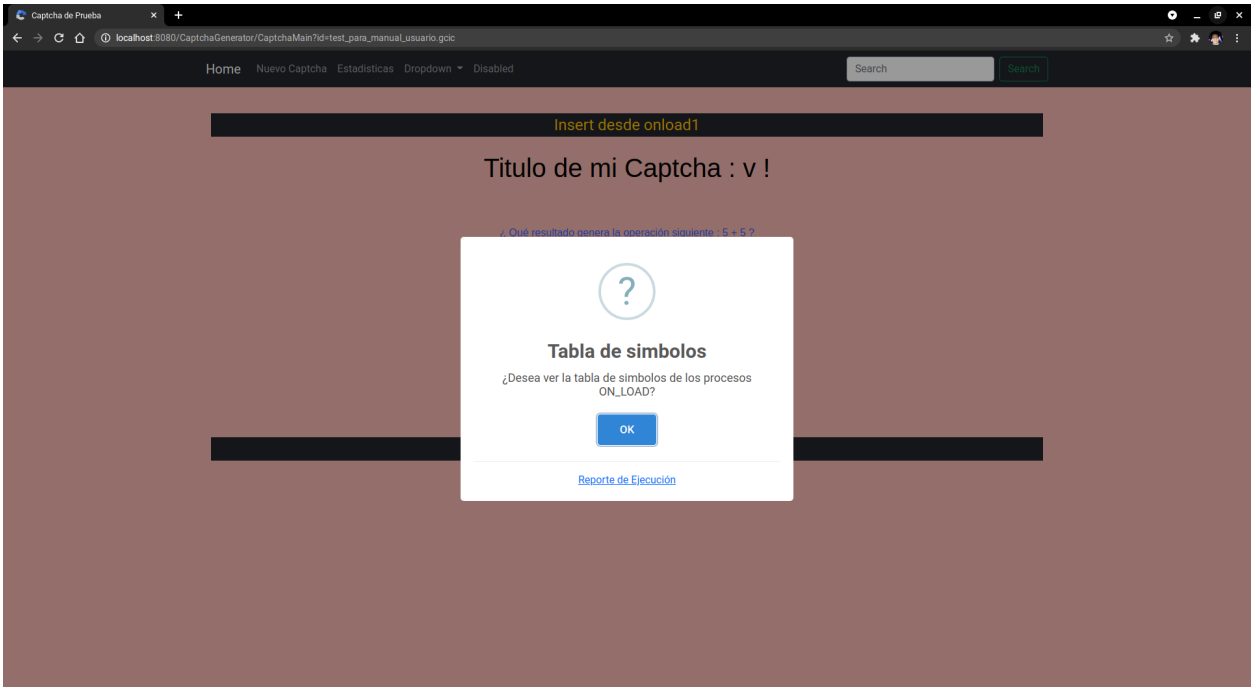
La tabla de errores, se encuentra junto al editor, de esta manera podremos ver los errores detectados en nuestro archivo, con una breve descripción, línea y columna y podemos proseguir a realizar las correcciones.

Una vez hechas las correcciones, hacemos clic en compilar y si todo está bien ahora tendremos la siguiente vista.



El mensaje indica que el captcha fue agregado y que ahora podremos hacer uso de él.

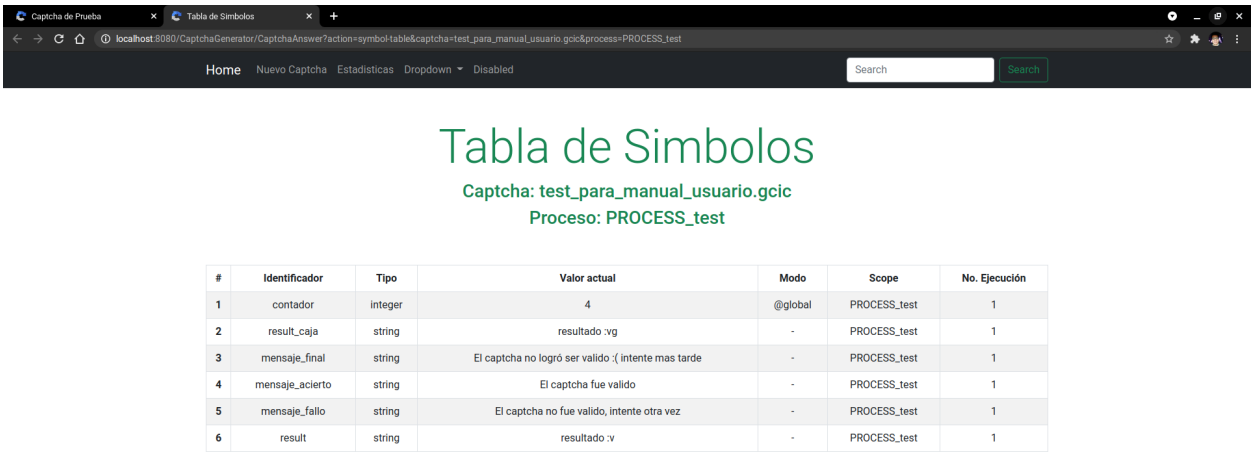
El captcha ya desplegado aparece de la siguiente manera, y la notificación nos indica que se ejecutaron procesos `ON_LOAD` y si deseamos, podemos ver la tabla de símbolos de las variables utilizadas en estos procesos.



Al responder o ejecutar las acciones solicitadas por el captcha y hacer clic en el botón disponible, se llama a la función o proceso, por ejemplo en la siguiente imagen ya se ejecuto el proceso y obtenemos la siguiente respueta:

En tal caso nos dice que la respuesta no es válida, también nos mostrará el siguiente mensaje para verificar la tabla de símbolos del proceso ejecutado.

Accediendo al reporte:



El reporte incluye id de la variable, tipo, valor actual, modo, scope de la variable y número de ejecución.

También se tiene un reporte de utilización, que muestra el nombre del captcha, enlace, número de intentos, número de aciertos y número de fallos.