MANUAL TÉCNICO

Información del sistema:

- • OS: Ubuntu 22.04.2 LTS
- • OS Type: 64-bit
- • CPU: Intel core i5 2.50 GHz
- • GPU: Mesa Intel HD Graphics 4400
- • Versión de java: 17
- • RAM: 8 GB
- • IDE: IntelliJ idea
- • Control de versiones: git 2.34.1
- • Github: https://github.com/Hatsune02/CaptchaForge.git

Descripción del Proyecto

Este proyecto consiste en una aplicación web desarrollada para la creación y evaluación de captchas embebidos, diseñada para un entorno seguro y flexible que permite a los usuarios generar captchas personalizados utilizando scripting. La arquitectura se basa en una combinación de frontend en Angular, una API en Java, y dos parsers especializados, uno para procesar el lenguaje embebido de captchas en el lado del servidor y otro para el procesamiento de expresiones en el frontend.

Tecnologías Utilizadas

1. Angular

Descripción: Angular es un framework de desarrollo frontend basado en TypeScript, que facilita la creación de aplicaciones de una sola página (SPA). Ofrece un sistema de componentes modular, lo que permite crear interfaces de usuario dinámicas y estructuradas. Además, Angular proporciona herramientas como su sistema de enrutamiento y servicios HTTP, que se utilizan para la comunicación con la API de backend.

Rol en el Proyecto: En este proyecto, Angular se encarga de la interfaz de usuario y permite a los usuarios interactuar de manera intuitiva con el sistema de captchas embebidos. Angular también se encarga de hacer llamadas a la API para obtener datos, enviar información, y actualizar la UI en tiempo real sin necesidad de recargar la página.

2. Java (API RESTful)

Descripción: Java es un lenguaje de programación de propósito general que ofrece robustez, seguridad y escalabilidad. Para este proyecto, Java se utilizó junto con un framework de desarrollo de API RESTful (como Spring Boot o JAX-RS) para crear el backend.

Rol en el Proyecto: La API desarrollada en Java es responsable de gestionar toda la lógica de negocio y la interacción con la base de datos. La API recibe solicitudes desde el frontend, valida y procesa la información, y devuelve las respuestas adecuadas. También gestiona el parser de captchas embebidos, que valida y evalúa el código escrito por los usuarios, y proporciona respuestas personalizadas.

3. Parser en JFlex y CUP

Descripción: JFlex y CUP son herramientas en Java para construir analizadores léxicos y sintácticos. JFlex se encarga del análisis léxico, generando un analizador que identifica y clasifica las cadenas de entrada, mientras que CUP (Constructor de Parsers para Universidades) crea un analizador sintáctico que interpreta las expresiones de acuerdo con una gramática definida.

Rol en el Proyecto: Este parser fue desarrollado para interpretar y validar el lenguaje embebido de captchas en el servidor. Se utiliza para evaluar el scripting en los captchas, verificando la sintaxis y las operaciones lógicas y matemáticas. Con JFlex y CUP, el servidor puede procesar y ejecutar el código de captchas de manera segura, permitiendo un flujo de trabajo dinámico y ajustado a los requisitos del proyecto.

4. Parser en Jison

Descripción: Jison es una herramienta de JavaScript que permite crear analizadores léxicos y sintácticos similares a Bison (una herramienta para C/C++). Jison es ampliamente utilizado para interpretar expresiones dentro de aplicaciones frontend y es compatible con el ecosistema de JavaScript.

Rol en el Proyecto: En el frontend, Jison se encarga del análisis de expresiones para evaluar y validar las entradas de usuario. Al integrar un parser con Jison en Angular, el sistema puede analizar las expresiones en tiempo real y dar retroalimentación inmediata, mejorando la experiencia de usuario y evitando errores de interpretación.

Gramatica

```
s ::= s_otp | c_body;

s_otp ::= cc c_scripting
      | c_scripting cc
      | cc
      | error body_labels
      ;

/* Etiquetas de apertura */
open_cc        ::= LT C_CC parameters_opt GT;
open_head      ::= LT C_HEAD GT;
open_body      ::= LT C_BODY parameters_opt GT;
open_title     ::= LT C_TITLE GT;
open_link      ::= LT C_LINK parameters_opt GT;
open_spam      ::= LT C_SPAM parameters_opt GT;
open_input     ::= LT C_INPUT parameters_opt GT;
open_textarea  ::= LT C_TEXTAREA parameters_opt C_CC GT;
open_select    ::= LT C_SELECT parameters_opt GT;
open_option    ::= LT C_OPTION GT;
open_div       ::= LT C_DIV parameters_opt GT;
open_img       ::= LT C_IMG parameters_opt GT;
open_br        ::= LT C_BR GT;
open_button    ::= LT C_BUTTON parameters_opt GT;
open_h1        ::= LT C_H1 parameters_opt GT;
open_p         ::= LT C_P parameters_opt GT;
open_scripting ::= LT C_SCRIPTING GT;

/* Etiquetas de cierre */
close_cc       ::= LT SLASH C_CC GT;
close_head     ::= LT SLASH C_HEAD GT;
close_body     ::= LT SLASH C_BODY GT;
close_title    ::= LT SLASH C_TITLE GT;
close_link     ::= LT SLASH C_LINK GT;
close_spam     ::= LT SLASH C_SPAM GT;
close_input    ::= LT SLASH C_INPUT GT;
close_textarea ::= LT SLASH C_TEXTAREA GT;
close_select   ::= LT SLASH C_SELECT GT;
close_option   ::= LT SLASH C_OPTION GT;
close_div      ::= LT SLASH C_DIV GT;
close_img      ::= LT SLASH C_IMG GT;
close_button   ::= LT SLASH C_BUTTON GT;
close_h1       ::= LT SLASH C_H1 GT;
close_p        ::= LT SLASH C_P GT;
close_scripting ::= LT SLASH C_SCRIPTING GT;
```

```
/* Definición de elementos */
cc ::= open_cc c_cc_body close_cc;

c_cc_body ::= c_head c_body;

c_head ::= open_head head_labels close_head
        | open_head close_head;

c_title ::= open_title string_text close_title;
c_link ::= open_link close_link;

head_labels ::= c_title c_link
            | c_link c_title
            | error;

c_body ::= open_body body_labels close_body
      | open_body close_body;

c_spam ::= open_spam body_labels close_spam
      | open_spam close_spam;

c_input ::= open_input body_labels close_input
      | open_input close_input;

c_textarea ::= open_textarea body_labels close_textarea
      | open_textarea close_textarea;

c_select ::= open_select body_labels close_select
      | open_select close_select;

c_option ::= open_option body_labels close_option
      | open_option close_option;

c_div ::= open_div body_labels close_div
      | open_div close_div;

c_img ::= open_img body_labels close_img
      | open_img close_img;

c_br ::= open_br;

c_button ::= open_button body_labels close_button
      | open_button close_button;

c_h1 ::= open_h1 body_labels close_h1
      | open_h1 close_h1;
```

```
c_p ::= open_p body_labels close_p
      | open_p close_p;

c_scripting ::= open_scripting global_stmt_opt close_scripting;

body_labels ::= body_label
         | body_labels body_label;

body_label ::= c_spam
             | c_input
             | c_textarea
             | c_select
             | c_option
             | c_div
             | c_img
             | c_br
             | c_button
             | c_h1
             | c_p
             | c_scripting
             | text
             | error;

parameters_opt ::=
             | parameters;

parameters ::= parameter
         | parameters parameter;

parameter ::= href
         | background
         | color
         | font_size
         | font_family
         | text_align
         | type
         | id
         | name
         | cols
         | rows
         | clase
         | src
         | width
         | height
         | alt
         | onclick
         ;
```

```
href ::= LBRACKET HREF EQUAL string_literal RBRACKET;

background ::= LBRACKET BACKGROUND EQUAL string_literal RBRACKET;

color ::= LBRACKET COLOR EQUAL string_literal RBRACKET;

font_size ::= LBRACKET FONT_SIZE EQUAL string_literal RBRACKET;

font_family ::= LBRACKET FONT_FAMILY EQUAL string_literal RBRACKET;

text_align ::= LBRACKET TEXT_ALIGN EQUAL string_literal RBRACKET;

type ::= LBRACKET TYPE EQUAL string_literal RBRACKET;

id ::= LBRACKET ID EQUAL string_literal RBRACKET;

name ::= LBRACKET NAME EQUAL string_literal RBRACKET;

cols ::= LBRACKET COLS EQUAL string_literal RBRACKET;

rows ::= LBRACKET ROWS EQUAL string_literal RBRACKET;

clase ::= LBRACKET CLASS EQUAL string_literal RBRACKET;

src ::= LBRACKET SRC EQUAL string_literal RBRACKET;

width ::= LBRACKET WIDTH EQUAL string_literal RBRACKET;

height ::= LBRACKET HEIGHT EQUAL string_literal RBRACKET;

alt ::= LBRACKET ALT EQUAL string_literal RBRACKET;

onclick ::= LBRACKET ONCLICK EQUAL string_literal RBRACKET;


global_stmt_opt ::=
            | global_stmt
            ;

global_stmt ::= method_declarator
            | global_stmt method_declarator
            ;

variable_declarators ::= variable_type GLOBAL variable_declarator
                  | variable_type variable_declarator
                  ;
```

```
variable_declarator ::= variable_id
                | variable_assignment_declarator
                ;

listIds ::= variable_id
          | listIds COMMA variable_id
          ;

variable_id ::= IDENTIFIER;

variable_assignment_declarator ::= listIds EQUAL expression;

variable_assignment ::= variable_id EQUAL expression;

variable_type ::= INTEGER
              | STRING
              | DECIMAL
              | CHAR
              | BOOLEAN
              ;

method_declarator ::= FUNCTION_ID LPAREN RPAREN method_block
                  | ON_LOAD LPAREN RPAREN method_block
                  ;

method_block ::= LBRACKET body_block_opt RBRACKET;

body_block_opt ::=
              | body_block
              ;

body_block ::= body_stmt
              | body_block body_stmt
              ;

body_stmt ::= variable_declarators SEMICOLON
          | variable_assignment SEMICOLON
          | function_call_stmt SEMICOLON
          | if_stmt
          | for_stmt
          | while_stmt
          | error
          ;

function_call_stmt ::= ALERT_INFO LPAREN expression RPAREN
                   | EXIT LPAREN RPAREN
```

```
            | REDIRECT LPAREN RPAREN
            | INSERT LPAREN params RPAREN
            ;

function_call_return ::= ASC LPAREN expression RPAREN
                | DESC LPAREN expression RPAREN
                | LETPAR_NUM LPAREN expression RPAREN
                | LETIMPAR_NUM LPAREN expression RPAREN
                | REVERSE LPAREN expression RPAREN
                | CARACTER_ALEATORIO LPAREN RPAREN
                | NUM_ALEATORIO LPAREN RPAREN
                | GET_ELEMENT_BY_ID LPAREN param RPAREN
                ;

params ::= param
      | params COMMA param
      ;

param ::= variable_id
      | simple_string
      ;

init_block ::= INIT LBRACE COLON
         ;

end_block ::= COLON RBRACE END
         ;

block_opt ::= init_block body_block_opt end_block
         ;

block ::= block_opt
      | block_stmt
      ;

block_stmt ::= variable_declarators SEMICOLON
         | variable_assignment SEMICOLON
         | function_call_stmt SEMICOLON
         ;

if_stmt ::= IF LPAREN expression RPAREN THEN block
         | IF LPAREN expression RPAREN THEN block else_if_stmts
         | IF LPAREN expression RPAREN THEN block else_stmt
         | IF LPAREN expression RPAREN THEN block else_if_stmts else_stmt
         ;

else_if_stmts ::= else_if_stmt
```

```
            | else_if_stmts else_if_stmt
            ;

else_if_stmt ::= ELSE IF LPAREN expression RPAREN THEN block
            ;

else_stmt ::= ELSE block
        ;

for_stmt ::= REPEAT LPAREN init_for RPAREN HUNTIL LPAREN expression
RPAREN block
        ;

init_for ::= INTEGER variable_id EQUAL expression
        | variable_id EQUAL expression
        ;

while_stmt ::= WHILE LPAREN expression RPAREN THENWHILE block
        ;

expression ::= conditional_expression
        ;

conditional_expression ::= conditional_or_expression
                ;

conditional_or_expression ::= conditional_and_expression
                    | conditional_or_expression OR conditional_and_expression
                    ;

conditional_and_expression ::= relational_expression
                | conditional_and_expression AND relational_expression
                ;

relational_expression ::= additive_expression
                | relational_expression REL_OP additive_expression
                ;

additive_expression ::= multiplicative_expression
                | additive_expression PLUS multiplicative_expression
                | additive_expression MINUS multiplicative_expression
                ;

multiplicative_expression ::= unary_expression
                    | multiplicative_expression TIMES unary_expression
                    | multiplicative_expression SLASH unary_expression
                    ;
```

unary_expression ::= value_literal
| variable_id
| function_call_return
| MINUS unary_expression
| LPAREN expression RPAREN
;

value_literal ::= string_literal
| NUMBER
| char_literal
| TRUE
| FALSE
;

string_literal ::= STRING_LITERAL
;

char_literal ::= CHAR_LITERAL
;

simple_string ::= SIMPLE_STRING;

---

a = [aA]
b = [bB]
c = [cC]
d = [dD]
e = [eE]
//f = [fF]
g = [gG]
h = [hH]
i = [iI]
//j = [jJ]
k = [kK]
l = [lL]
m = [mM]
n = [nN]
o = [oO]
p = [pP]
//q = [qQ]
r = [rR]

```
s = [sS]
t = [tT]
u = [uU]
v = [vV]
//w = [wW]
x = [xX]
y = [yY]
//z = [zZ]

LineTerminator = \r|\n|\r\n
WhiteSpace = {LineTerminator} | [ \t\f]

/* ___Reserved words___ */

   /* CC labels */
c_cc = {c}_{c}{c}
c_head = {c}_{h}{e}{a}{d}
c_title = {c}_{t}{i}{t}{l}{e}
c_link = {c}_{l}{i}{n}{k}
c_body = {c}_{b}{o}{d}{y}
c_spam = {c}_{s}{p}{a}{m}
c_input = {c}_{i}{n}{p}{u}{t}
c_textarea = {c}_{t}{e}{x}{t}{a}{r}{e}{a}
c_select = {c}_{s}{e}{l}{e}{c}{t}
c_option = {c}_{o}{p}{t}{i}{o}{n}
c_div = {c}_{d}{i}{v}
c_img = {c}_{i}{m}{g}
c_br = {c}_{b}{r}
c_button = {c}_{b}{u}{t}{t}{o}{n}
c_h1 = {c}_{h}1
c_p = {c}_{p}
c_scripting = {c}_{s}{c}{r}{i}{p}{t}{i}{n}{g}

   /* Parameters */
href = (href)
background = (background)
color = (color)
font_size = (font-size)
font_family = (font-family)
text_align = (text-align)
type = (type)
id = (id)
name = (name)
cols = (cols)
rows = (rows)
class = (class)
src = (src)
```

width = (width)
height = (height)
alt = (alt)
onclick = (onclick)

/* Data type */
integer = (integer)
decimal = (decimal)
boolean = (boolean)
char = (char)
string = (string)

/* Functions */
asc = (ASC)
desc = (DESC)
letpar_num = (LETPAR_NUM)
letimpar_num = (LETIMPAR_NUM)
reverse = (REVERSE)
caracter_aleatorio = (CARACTER_ALEATORIO)
num_aleatorio = (NUM_ALEATORIO)
alert_info = (ALERT_INFO)
exit = (EXIT)
redirect = (REDIRECT)
insert = (INSERT)
getElemenById = (getElemenById)
on_load = (ON_LOAD)
/* Mode */
global = (@global)

/* Function Block */
init = (INIT)
end = (END)
if = (IF)
then = (THEN)
else = (ELSE)
repeat = (REPEAT)
huntil = (HUNTIL)
while = (WHILE)
thenwhile = (THENWHILE)

/* Booleans */
true = (true)
false = (False)

/* Operators */
Plus = [+]
Minus = [-]

```
Times = [*]
Division = [\/]
LParen = [\(]
RParen = [\)]

relatedOperations = "==" | "!=" | "<=" | ">="
or = "||"
and = "&&"
not = "!"


/* Structures */

LBracket = [\[]
RBracket = [\]]
LBrace = [\{]
RBrace = [\}]
LThan = [<]
GThan = [>]
Colon = [:]
Semicolon = [;]
Comma = [,]
VerticalBar = [|]

/* Double String */
Q = [\"]
StringContent = [^\\\n\"]*
String = {Q}{StringContent}{Q}
/* Simple String */
QSingle = \'
StringContentSingle = ([^\'\\])([^\'\\])+
StringSingle = {QSingle}{StringContentSingle}{QSingle}
one_char = {QSingle}[^\'\\]{QSingle}

/* Numbers */
digit = [0-9]+
decinteger = (([1-9]{digit}*) | "0")
number = {decinteger}
double = {decinteger}("."{digit}?{digit}?{digit}?{digit}?)?
equal = "="

/* Comments */
init_multiline_comment =  "<!--"
end_multiline_comment = "-->"
init_inline_comment = "!!"
inline_comment = {init_inline_comment}[^\n]*
multiline_comment = {init_multiline_comment}[^]*{end_multiline_comment}
```

comment = {inline_comment} | {multiline_comment}

/* Identifier */
letter = [a-zA-Z]
identifier = {letter}({letter}|{digit}|_)*
text = [_`.\|a-zA-Z0-9áéíóú¿?‘’“”¡@]|-_.\/`\|a-zA-Z0-9áéíóú¿?‘’“”¡@]*
function_id = "FUNCTION_"{identifier}