**Instituto Tecnológico y de Estudios Superiores de Monterrey**

**Campus Puebla**

# Implementación de Métodos Computacionales

**TC2037.601**

## Evidencia 1. Resaltador de sintaxis

**Profesora: Dr. Luciano García Bañuelos**


**Estudiantes:**

**Daniel Francisco Acosta Vázquez / A01736279**

**Diego García de los Salmones Ajuria / A01736106**

**Raúl Díaz Romero / A01735839**

**Rogelio Hernández Cortés / A01735819**


**Periodo Febrero - Junio 2023**

**5 / Mayo / 2023**

# Evidencia 1. Resaltador de sintaxis

## 1. Link al Repositorio en GitHub

Link: https://github.com/RoloDiaz/Resaltador-sintaxis.git

Igualmente se le envió una invitación al repositorio a través de su correo electrónico, en caso de no visualizar dicha invitación favor de mandarnos un mensaje para enviarla de nuevo.

## 2. Instrucciones para probar la Solución

Primero, debe de existir un archivo de JavaScript en la carpeta donde se encuentra el código.

Posteriormente, se debe de escribir el siguiente comando en la terminal:

*escript evi1 pruebaActual.js resultado.html*

El anterior comando contiene múltiples aspectos importantes tales como:

- **escript**; Palabra que funciona para ejecutar el código.
- **evi1**; Nombre del módulo que se desea ejecutar.
- **pruebaActual.js**; Archivo de entrada el cual se analizará y debe existir en la carpeta. Si se encuentra dentro de una carpeta dentro de la carpeta principal se accede a él escribiendo en el comando NombredelaCarpeta/pruebaActual.js en vez de únicamente pruebaActual.js.
- **resultado.html**; Archivo en formato html (se puede visualizar en el buscador) que se creará, en el cual se incluirá todo el código resaltado del archivo de entrada.

## 3. Reflexión

**Solución Planteada**

El proyecto de elaboración de un resaltador de sintaxis en Elixir para JavaScript es una muestra del potencial que tienen las habilidades y conocimientos adquiridos durante el estudio de la programación en Elixir y Erlang. En este proyecto, no solo demostramos la capacidad de aplicar los conceptos teóricos aprendidos, sino que también mostramos la capacidad de innovar y crear soluciones prácticas con base en lo aprendido en este curso.

La creación de un resaltador de sintaxis en Elixir para JavaScript fue una tarea que requirió conocimiento y comprensión de ambos lenguajes de programación, pues fue sumamente necesario conocer las particularidades y características únicas de cada uno de los dos lenguajes para poder crear una herramienta que funcione de manera efectiva. Además, se

requirió habilidad para identificar los patrones y estructuras de código que deben ser resaltados para mejorar la legibilidad y la comprensión del mismo.

**Categorías léxicas avanzadas omitidas**

- **String Interpolation**, esta categoría léxica es el proceso de evaluar una cadena/string que contiene una o más variables para luego reemplazar a estas variables con sus valores correspondientes. Esta categoría Léxica se omitió debido a que se debería hacer un escaner con soporte a varios niveles (lo cual el LEEX de Erlang no soporta).
- **Unicode Character,** esta categoría léxica implica el uso de caracteres no incluidos en el código ASCII, por ejemplo: '你好' o '\u4f60\u597d'. Se decidió omitir parcialmente está categoría debido a que el LEEX de Erlang no suele soportar caracteres fuera del código ASCII, además, el manejo de los Unicode Characters puede llegar a ser muy complejo, ya que se pueden utilizar como identificadores.

  Por último, esta categoría se omitió parcialmente debido a que los Unicode Characters son reconocidos dentro de strings (‘’,””,``), solamente no son reconocidos al ser utilizados como identificadores.
- **Diferenciación entre resaltados,** esta categoría léxica se refiere al cómo ciertos resaltadores de sintaxis resaltan de manera diferente determinadas palabras reservadas o identificadores dependiendo el contexto, por ejemplo, algunos resaltadores diferencían el resaltado del nombre de una variable normal con el nombre de una función o una clase, esto a pesar de que todos los nombres son esencialmente identificadores. Otro ejemplo de esto es como algunos resaltadores diferencían el resaltado entre atributos y métodos de una clase.

  Por cuestiones prácticas se decidió ignorar estas diferenciaciones y resaltar todas las palabras reservadas de un mismo color y los identificadores de otro color, independientemente del contexto en que se encuentren estas.

**Algoritmos, Tiempos de Ejecución y Complejidad**

Los algoritmos utilizados en el resaltador de sintaxis de Elixir para JavaScript se basan en el análisis léxico de ambos lenguajes. Para resaltar correctamente la sintaxis de un código en JavaScript, el resaltador necesita identificar las palabras clave, las variables, los literales y otros elementos importantes en el código. Para hacer esto, el resaltador utiliza una combinación de expresiones regulares, las cuales implican la identificación de las palabras clave y los símbolos importantes en el código, como los operadores y los paréntesis.

Antes de detallar la complejidad del programa, hay que recordar que el escáner del Resaltador de Sintaxis se implementó con la herramienta 'LEEX' del lenguaje de

programación 'Erlang'. Por ello, dicho escáner funciona prácticamente igual a un autómata o una máquina de estados, además, los componentes de dicho autómata están conformados por las expresiones regulares establecidas en las reglas de sintaxis del escáner.
Tomando en cuenta lo anterior, el programa comienza el análisis al recibir el archivo de entrada (archivo de JavaScript), esté lee el archivo y lo procesa letra por letra.
Cada letra induce una transición en el autómata generado por el escáner, esto para detectar a qué regla pertenece, una vez se identifica la regla a la que pertenece la letra o palabra analizada, se le asigna un token/id que representa a qué regla léxica pertenece. Posteriormente, dependiendo del token/id de cada elemento del archivo, se genera código html para resaltar las letras o palabras analizadas.
Por último, se genera un archivo .html que incluye todo el código del archivo de entrada resaltado correctamente.
En conclusión, la complejidad del Resaltador de sintaxis es lineal (O(n)), pues en el peor de los casos se genera un token/id por cada letra analizada.
En cuanto al tiempo de ejecución dependerá del tamaño del código que se esté resaltando y del hardware utilizado, para un código pequeño y simple, el tiempo de ejecución es relativamente rápido; casi al instante, sin embargo, para códigos más grandes y complejos, el tiempo de ejecución llega a ser significativamente mayor, consumiendo unos cuantos segundos.

## 4. Implicaciones Éticas

**Diego García de los Salmones Ajuria**

Un resaltador de sintaxis, como cualquier tecnología, tiene implicaciones éticas que deben ser consideradas cuidadosamente, por ello, es importante asegurar que su uso tenga un impacto positivo en la sociedad mediante los siguientes aspectos:

- **Sesgo en la programación**: El resaltador de sintaxis puede ser programado para resaltar ciertos elementos de código más que otros, lo que podría generar un sesgo involuntario en la interpretación del código, por lo que es importante tener en cuenta que el resaltador debe ser programado de manera imparcial para evitar cualquier tipo de discriminación o sesgo.
- **Privacidad y seguridad**: El resaltador de sintaxis puede leer y analizar el código de los usuarios, lo que puede plantear preocupaciones respecto a la privacidad y seguridad de la información, por lo que es importante asegurarse de que se implementen medidas adecuadas para proteger la información confidencial de los usuarios.

- **Automatización de la programación**: La creación de herramientas que facilitan la programación puede tener implicaciones éticas en cuanto a la automatización del trabajo de los programadores así como implicaciones dentro del mercado laboral.
- **Responsabilidad del programador**: La utilización del resaltador de sintaxis puede hacer que los programadores confíen demasiado en las herramientas y no presten suficiente atención a la comprensión profunda del código que están escribiendo.

**Raúl Díaz Romero:**

Una de las principales tecnologías utilizadas fue la herramienta 'LEEX' de Erlang, dicha herramienta utiliza expresiones regulares para manipular e identificar patrones en textos.

Entonces, las expresiones regulares pueden ser utilizadas para dar un impacto positivo o negativo en nuestra sociedad tanto a programadores como usuarios normales.

Algunos ejemplos de sus aplicaciones son:

**Impacto Benéfico:**

1. **Validación de datos:** Las expresiones regulares pueden ser utilizadas para validar que los datos ingresados por los usuarios cumplan con un formato específico. Por ejemplo, se pueden utilizar para validar que un correo electrónico tenga un formato válido o que una contraseña sea lo suficientemente segura en una página registro o inicio de sesión.
2. **Búsqueda de información:** Las expresiones regulares pueden ser utilizadas para buscar información específica en grandes cantidades de texto.

**Impacto Negativo:**

1. **Ataques de denegación de servicio (DoS):** Las expresiones regulares complejas y mal diseñadas pueden causar ataques de denegación de servicio en servidores web. Los atacantes pueden enviar solicitudes de búsqueda que contengan expresiones regulares maliciosas que consuman recursos del servidor y lo hagan inaccesible.
2. **Monitoreo de actividad:** Las expresiones regulares pueden ser utilizadas para monitorear la actividad en línea de los usuarios sin su consentimiento. Por ejemplo, se pueden utilizar para capturar información de tarjetas de crédito en sitios web no seguros.

En resumen, las expresiones regulares, al igual que cualquier tipo de tecnología utilizada hoy en día, pueden llegar a tener un impacto positivo o negativo en nuestra sociedad, esto depende directamente del contexto y propósito en el que se utilicen

**Daniel Francisco Acosta Vázquez:**

El utilizar tecnologías como 'LEEX' de Erlang nos abre las posibilidades de utilizar expresiones regulares para diferentes ámbitos de la tecnología. Muchos de estos ámbitos sin dudas utilizan esta tecnología para mejoras en la sociedad pero hay que tener en cuenta las implicaciones éticas de esta tecnología. En retrospectiva, el uso de expresiones regulares puede conllevar a aplicaciones maliciosas, como por ejemplo la recopilación de información de usuarios sin su consentimiento, como por ejemplo contraseñas o información sensible. También, se podrían utilizar en la elaboración de ataques de denegación de servicio o para romper el código de alguna aplicación con intenciones maliciosas. Sin embargo, hay que tener en cuenta que hay muchas posibles aplicaciones positivas para esta tecnología, tal como la automatización o el arreglo automatizado de códigos, al igual que la validación y análisis de datos como grandes herramientas. Podemos concluir que estas tecnologías, con un uso ético y apropiado pueden ayudar en el desarrollo de nuevas tecnologías para nuestra sociedad y que por ende se debe entender su funcionamiento para poder aplicarlos a futuros proyectos.

**Rogelio Hernández Cortés:**

Teniendo en cuenta que una de las tecnologías que hicieron posible este proyecto fue LEEX de Erlang,que utiliza expresiones regulares para analizar textos y la cual tiene distintas implicaciones éticas.

Por el lado de las positivas, es que puede servir para realizar filtros de información, esto al ser aplicado en áreas de ciberseguridad como podría ser la identificación de posibles amenazas en campos de entrada que pretendan realizar inyección de código o también en áreas de análisis de datos, al disminuir el tiempo en el que se analiza la información.

Por el lado de las negativas, se pueden utilizar las expresiones regulares para bloquear navegadores web, bloquear firewall de aplicaciones web o incluso atacar directamente bases de datos, esto se conoce como ReDOS, que es una denegación de servicios por expresiones regulares.

De esta manera podemos concluir que las expresiones regulares tiene un potencial muy grande que puede ser benéfico o negativo para la sociedad, por eso mismo debemos seguir conociendo aún más sobre estos tipos de tecnología e intentar buscar soluciones a las malas prácticas que estas pudieran tener.