Instituto Tecnológico de Estudios Superiores de Monterrey

TC2037.601

Implementación de Métodos Computacionales

Actividad Integradora 3.4 Resaltador de sintaxis (evidencia de competencia)

Juan Carlos Ferrer Echeverría A0173494

Fecha de entrega: 23/04/22

Con esta actividad integradora aprendimos a implementar código en múltiples lenguajes como Elixir, Erlang, documentado con HTML y formato en CSS con el propósito de generar un resaltador léxico de un lenguaje de programación de nuestra preferencia, que en caso efectivo de este estudio fue Python. Por otro lado, aprendimos a identificar y clasificar los objetos léxicos de un lenguaje de programación, además de realizar una investigación extensiva para cubrir todos los objetos básicos que componen en su mayoría la estructura de un script de Python.

Para realizar esta actividad utilizamos expresiones regulares con lenguaje Erlang para clasificar cada una de las literales, símbolos y palabras claves que componen Python, además de asignarle una regla o categoría gramatical en nuestra estructura. Posteriormente utilizamos Elixir para manejar los resultados de la entrada de nuestro script en lenguaje Python por nuestro Lexer en Erlang, darle a cada categoría una clase de formato CSS y generar el archivo HTML. Debido a que el documento HTML se generar de forma automática, debemos generar un archivo CSS para darle formato a cada una de nuestras categorías para que sean identificables a simple vista. Entre las categorías léxicas avanzadas que decidimos dejar fuera de nuestro script o que fueron añadidas a alguna de nuestras categorías básicas se encuentran las siguientes:

- Indentación propia
- Literales de bytes
- Cadenas de texto formateadas
- Uniones de línea explícita e implícita

Tomamos esta decisión ya sea porque su sintaxis léxica no era posible de implementar por interpretación propia de las expresiones regulares del código en Erlang, o porque para el propósito de este proyecto, que es el resaltar el comportamiento léxico de las palabras, entraban en otras categorías básicas, como es el caso de los literales de bytes. En este caso, fue efectivo tomar esta decisión para resaltar su sintaxis léxica, ya que, en posteriores avances, donde se estudie la gramática y su comportamiento en el lenguaje (Python), si será necesario realizar esta diferenciación de cada categoría en nuestro script de estas clases léxicas avanzadas.

Por otro lado, el impacto léxico que puede tener esta tecnología en la sociedad puede ser de forma positiva, ya que se puede agilizar la búsqueda de elementos de una categoría léxica seleccionada de forma rápida y de forma académica puede ser útil para diseñar la forma en la que queremos que se diferencien los elementos de un script de algún lenguaje para que sea más amigable y apreciable su presentación, lo que agiliza en gran medida su entendimiento. En cuanto a la complejidad de nuestro algoritmo, la razón es de forma lineal, ya que el mismo número de elementos introducidos en nuestro script de Python es el mismo número de elementos que tiene que procesar y que expulsa nuestro Lexer en forma de Tokens de acuerdo con su categoría. Por último, se puede afirmar que esta complejidad está relacionada con el tiempo de ejecución de nuestro algoritmo, ya que utilizamos la herramienta time para comprobarlo. Mientras más elementos contenga nuestro script, más tiempo le llevará procesarlo a nuestro algoritmo.