TECNOLÓGICO DE MONTERREY

CAMPUS MONTERREY

Actividad 6.1 Análisis de herramientas usadas para la solución de las situaciones problema

Fernando Daniel Monroy Sánchez A01750536

Eugenio Andrés Mejía Fanjón A01412143

TC2037. Implementación de métodos computacionales

Docentes:

Jesús Guillermo Falcón Cardona

Domingo, 9 de Junio del 2024

Durante el presente curso se desarrollaron múltiples actividades con las que se reforzaron conocimientos en teoría de la computación, programación funcional y programación concurrente. Al inicio del semestre se introdujeron los autómatas finitos deterministas y no deterministas, estructuras fundamentales para el entendimiento de las bases de la computación. Le siguieron los conceptos de expresiones regulares y gramáticas libres de contexto, para finalizar con máquinas de Turing, una representación abstracta pero inherente a las computadoras como las conocemos hoy en día.

La primera evidencia que se desarrolló fue un resaltador de sintaxis para el lenguaje de programación Python. En el programa desarrollado, se utilizaron conceptos de expresiones regulares para realizar el *parsing* de los tokens que conforman la sintaxis del programa y desplegarlos a través de códigos de colores a través de documentos HTML y estilos con CSS. El proceso de investigación a fondo de la documentación del lenguaje nos permitió obtener una mayor perspectiva y entendimiento sobre cómo los compiladores y los intérpretes procesan el código fuente.

Para mejorar la implementación de la solución anterior se propone utilizar *frameworks* y *librerías* externas, desarrolladas por terceros y no directamente por la organización detrás del lenguaje de programación, para mejorar y facilitar el proceso. Esto se alinea con una de las bases en el desarrollo de software, que es la colaboración globalizada con el software abierto. El objetivo es que las personas que se encuentran con un problema o que buscan implementar una solución puedan concretarlas fácilmente a través de utilidades mantenidas por desarrolladores anteriores.

Una librería altamente funcional para solucionar este problema es Highlight.js, usada con JavaScript. Esta librería es compatible con más de 190 lenguajes y tiene disponibles más de 400 temas diferentes para la visualización sintáctica en tiempo real de código. Otra opción para dar solución al problema es Pygments, que se importa como un paquete de Python. Tiene opciones para introducir código de múltiples lenguajes así como opciones para exportar el resaltado hacia formatos comunes como HTML, LaTeX, RTF y más.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <link rel="stylesheet" href="/path/to/styles/default.min.css">
    <script src="/path/to/highlight.min.js"></script>
    <script>hljs.highlightAll();</script>
</head>
<body>
    <h1>Ejemplo de Resaltado de Sintaxis con Highlight.js</h1>
    <pre><code class="language-python">
```

```
# Ejemplo de código Python
def saludo(nombre):
    print(f"Hola, {nombre}!")
saludo("Mundo")
```

```
    </code></pre>
</body>
</html>
```

Propuesta de uso de librería Highlight.js

```
from pygments import highlight
from pygments.lexers import PythonLexer
from pygments.formatters import HtmlFormatter

# Código Python a resaltar
code = '''
# Ejemplo de código Python
def saludo(nombre):
    print(f"Hola, {nombre}!")

saludo("Mundo")
'''

# Aplicar resaltado de sintaxis
formatter = HtmlFormatter(full=True, linenos=True, style='colorful')
highlighted_code = highlight(code, PythonLexer(), formatter)

# Guardar el resultado en un archivo HTML
with open('highlighted_code.html', 'w') as file:
    file.write(highlighted_code)

print("El código resaltado se ha guardado en 'highlighted_code.html'")
```

Propuesta de uso de librería Pygments

Ambas librerías logran dar solución al problema, sin embargo, tomando en cuenta la facilidad de implementación como métrica, la solución con Highlight.js es mejor, pues con simple código HTML y especificando el lenguaje de programación al cuál se le hará sintaxis, es fácil generar resaltar la sintaxis.

Por otro lado, se programó un sistema bancario en Golang que aprovecha la concurrencia para realizar una mayor cantidad de transacciones en un menor tiempo, así haciendo del sistema más eficiente y resistente a picos de uso. Además, se implementaron de manera correcta estándares para asegurar la integridad de los datos tal como *mutex* con el objetivo de evitar condiciones de carrera y deadlocks. A través del desarrollo de esta evidencia pudimos comprender más los beneficios de sistemas distribuidos cuando se ejecutan correctamente.

Existen diferentes tecnologías con las que se puede implementar el anterior sistema de igual manera. La primera de estas es *Elixir*, un lenguaje de programación funcional y concurrente. Es un lenguaje ideal ya que está diseñado para aplicaciones concurrentes y tiene un diseño orientado hacia procesos ligeros y de muchas conexiones simultáneas. La segunda tecnología con la que se podría implementar la solución es *Scala*, otro lenguaje de programación funcional que, al usarse en conjunto con el toolkit *Akka*, permite construir aplicaciones distribuidas. Sus principales ventajas es que utiliza un modelo de actores para encapsular el estado de cada uno por separado y se ejecuta sobre una JVM (Java Virtual Machine).

```elixir
defmodule Banco do
  def deposit(balance, amount) do
    balance + amount
  end

  def withdraw(balance, amount) do
    balance - amount
  end
end

# Uso del módulo Banco
balance = 1000
balance = Banco.deposit(balance, 500)
balance = Banco.withdraw(balance, 200)
IO.puts("El balance es: #{balance}")
```

Propuesta de implementación en Elixir

```scala
object BankAccount {
  def deposit(balance: Int, amount: Int): Int = balance + amount

  def withdraw(balance: Int, amount: Int): Int = balance - amount
}

object BankSystem extends App {
```

```
  var balance = 1000
  balance = BankAccount.deposit(balance, 100)
  balance = BankAccount.withdraw(balance, 50)

  println(s"El balance es: $balance")
}
```

Propuesta de implementación en Scala

Para finalizar, existen múltiples tecnologías emergentes debido a diferentes factores que han tenido una mayor relevancia en los últimos años, tales como la necesidad de escalabilidad, para el manejo de una mayor cantidad de usuarios y peticiones simultáneamente, la evolución del hardware, con nuevos avances en procesadores multinúcleo y mayor demanda por parte de frameworks para aprovechar estos avances, e innovación en la industria por gigantes como Google y Facebook.

A partir de estas tecnologías emergentes y similares es que se ha logrado satisfacer nuevas demandas, nuevos problemas específicos, y se ha tenido una mayor colaboración.

Para concluir, el pensamiento formal tiene un rol muy importante en el entendimiento de nuevas tecnologías, pues proporcionan una base sólida para su comprensión. Un gran ejemplo de esto es la máquina de Turing, sobre la cual se han desarrollado compiladores y lenguajes de programación de alta fiabilidad. Además, considero que a través de esta educación sobre los fundamentos de computación se puede seguir innovando sobre los mismos pilares que ya proveen una clara estructura en la manera en que se pueden abordar nuevos descubrimientos.