Informe de Laboratorio 06

Tema: Tries

Nota

Estudiantes	Docente	Asignatura
Florez Bailon Luis Fernando Champi Sanchez Manuel Mario	Mg. Edith Giovanna Cano Mamani	Laboratorio EDA Semestre: III Código: 1702122

Laboratorio	Tema	Duración
06	Tries	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - A	Del 19 Julio 2023	Al 23 Julio 2023

1. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/ManuelChampi/Eda_Lab_D23A`
- URL para el laboratorio 06 en el Repositorio GitHub.
- `https://github.com/ManuelChampi/Eda_Lab_D23A/tree/main/Eda_Lab_06`

2. Actividades con el repositorio GitHub

2.1. Resolución de ejercicio

2.1.1. Clase TrieGUI

- Se crea la interfaz gráfica de usuario.

```java
public class TrieGUI extends JFrame {
    private Trie trie;
    private JTextField textField;
    private JTextArea textArea;
    private JButton insertButton;
    private JButton searchButton;
    private JButton replaceButton;

    public TrieGUI() {
        // Configuración de la ventana
        setTitle("Trie GUI");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(800, 400); // Tamaño inicial de la ventana
        setLocationRelativeTo(null); // Centrar la ventana en la pantalla

        // Inicializar Trie
        trie = new Trie();

        // Inicializar componentes
        textField = new JTextField();
        textArea = new JTextArea();
        textArea.setEditable(false); // Bloquear el cuadro de texto para no poder editarlo manualmente
        insertButton = new JButton("Insertar");
        searchButton = new JButton("Buscar");
        replaceButton = new JButton("Reemplazar");

        // Agregar componentes a la ventana
        JPanel inputPanel = new JPanel(new BorderLayout());
        inputPanel.add(textField, BorderLayout.NORTH);
        inputPanel.add(new JScrollPane(textArea), BorderLayout.CENTER);
        add(inputPanel, BorderLayout.CENTER);

        JPanel buttonPanel = new JPanel(new FlowLayout());
        buttonPanel.add(insertButton);
        buttonPanel.add(searchButton);
        buttonPanel.add(replaceButton);
        add(buttonPanel, BorderLayout.SOUTH);
```

Figura 1: Constructor TriGUI

- Creamos los respectivos listeners para los botones insertar, buscar y reemplazar.

```
47          insertButton.addActionListener(new ActionListener() {
48              @Override
49              public void actionPerformed(ActionEvent e) {
50                  insertWord();
51              }
52          });
53
54          searchButton.addActionListener(new ActionListener() {
55              @Override
56              public void actionPerformed(ActionEvent e) {
57                  searchWord();
58              }
59          });
60
61          replaceButton.addActionListener(new ActionListener() {
62              @Override
63              public void actionPerformed(ActionEvent e) {
64                  replaceWord();
65              }
66          });
67      }
```

Figura 2: Listeners para los botones

- El método insertWord nos permitirá ingresar el texto que el usuario desee.

```
69      private void insertWord() {
70          String word = textField.getText().trim();
71          if (!word.isEmpty()) {
72              trie.insert(word);
73              textField.setText("");
74              updateTextArea();
75          }
76      }
```

Figura 3: Metodo insertWord

- El método searchWord nos permite buscar una palabra específica que sea ingresada por el usuario. Si la palabra no está vacía, busca la cantidad de apariciones de esa palabra en el texto utilizando indexOf y muestra un mensaje con el resultado.

```
78    private void searchWord() {
79        String word = textField.getText().trim();
80        if (!word.isEmpty()) {
81            int count = trie.search(word);
82            String message = "La palabra '" + word + (count > 0 ? "' se encuentra "
83            + count + " veces en el texto." : "' no se encuentra en el texto.");
84            JOptionPane.showMessageDialog(this, message);
85        }
86    }
```

Figura 4: Metodo searchWord

- El método replaceWord nos pide ingresar una palabra, luego el programa buscará todas las palabras iguales en el texto y las reemplaza por la palabra que indiquemos. Luego actualiza el área de texto para mostrar el texto modificado.

```
88    private void replaceWord() {
89        String word = textField.getText().trim();
90        String replacement = JOptionPane.showInputDialog(this, "Ingrese la palabra de reemplazo:");
91        if (!word.isEmpty() && replacement != null) {
92            int count = trie.replace(word, replacement);
93            String message = "Se reemplazaron " + count +
94                    " apariciones de la palabra '" + word + "' con '" + replacement + "'.";
95            JOptionPane.showMessageDialog(this, message);
96            textField.setText("");
97            updateTextArea();
98        }
99    }
```

Figura 5: Metodo replaceWord

- Método para actualizar el área de texto con el texto almacenado en la Trie. Formatea el texto almacenado en la Trie para ajustarlo al ancho del área de texto, realizando un salto de línea cuando el texto supere el ancho de la ventana.

```
101    private void updateTextArea() {
102        String formattedText = formatText(trie.getText(), textArea.getWidth());
103        textArea.setText(formattedText);
104    }
105
```

Figura 6: Metodo updateTextArea

- Método para formatear el texto para ajustarlo al ancho del área de texto.

```
106     private String formatText(String text, int width) {
107         StringBuilder formattedText = new StringBuilder();
108         String[] words = text.split(" ");
109         int lineLength = 0;
110 
111         for (String word : words) {
112             if (lineLength + word.length() <= width) {
113                 formattedText.append(word).append(" ");
114                 lineLength += word.length() + 1;
115             } else {
116                 formattedText.append("\n").append(word).append(" ");
117                 lineLength = word.length() + 1;
118             }
119         }
120 
121         return formattedText.toString();
122     }
```

Figura 7: Metodo formatText

- Se crea el main para poder correr el programa.

```
124     public static void main(String[] args) {
125         SwingUtilities.invokeLater(new Runnable() {
126             @Override
127             public void run() {
128                 new TrieGUI().setVisible(true);
129             }
130         });
131     }
132 }
```

Figura 8: Main

2.1.2. Clase Trie

- Clase que representa la estructura de datos Trie. Tiene un nodo raíz y un StringBuilder para almacenar el texto ingresado. Permite insertar palabras en la Trie, realizar búsqueda de palabras y reemplazar palabras en el texto..

```
6  public class Trie {
7      private TrieNode root;
8      private StringBuilder text;
9
10     public Trie() {
11         root = new TrieNode();
12         text = new StringBuilder();
13     }
```

Figura 9: constructor Trie

- "insert(String word)" es un método para insertar una palabra en la Trie. Recibe una palabra como parámetro y la inserta en la Trie. Comienza en el nodo raíz y recorre cada carácter de la palabra. Para cada carácter, verifica si ya existe un nodo hijo con ese carácter en el nodo actual. Si no existe, crea un nuevo nodo hijo con ese carácter. Luego, avanza al nodo hijo correspondiente. Una vez que ha recorrido todos los caracteres de la palabra, marca el último nodo como el final de una palabra (isEndOfWord se establece en true) y agrega la palabra al StringBuilder text, seguida de un espacio en blanco.

```
15     public void insert(String word) {
16         TrieNode node = root;
17         for (char c : word.toCharArray()) {
18             node.children.putIfAbsent(c, new TrieNode());
19             node = node.children.get(c);
20         }
21         node.isEndOfWord = true;
22         text.append(word).append(" ");
23     }
```

Figura 10: Metodo insert

- "search(String word)" es un método para buscar una palabra en la Trie. Recibe una palabra como parámetro y busca la cantidad de apariciones de esa palabra en el texto almacenado en la Trie. Utiliza el método indexOf del StringBuilder text para buscar la palabra en el texto. Inicialmente, count se establece en 0. Mientras encuentre la palabra en el texto (indexOf devuelve un índice válido), incrementa count y busca la siguiente aparición de la palabra a partir del índice siguiente. El método devuelve la cantidad de apariciones encontradas.

```
25      public int search(String word) {
26          int count = 0;
27          int index = text.indexOf(word + " ");
28          while (index != -1) {
29              count++;
30              index = text.indexOf(word + " ", index + 1);
31          }
32          return count;
33      }
```

Figura 11: Metodo search

- "replace(String word, String replacement)" es un método para reemplazar una palabra en la Trie. Recibe dos parámetros: word es la palabra que se va a reemplazar y replacement es la palabra de reemplazo. Busca la primera aparición de word en el texto utilizando indexOf. Mientras encuentre word en el texto (indexOf devuelve un índice válido), reemplaza la palabra con replacement utilizando el método replace del StringBuilder. Incrementa count para mantener un registro de cuántos reemplazos se han realizado. Luego, busca la siguiente aparición de word a partir del índice siguiente al reemplazo realizado. El método devuelve la cantidad de reemplazos realizados.

```
35      public int replace(String word, String replacement) {
36          int count = 0;
37          int index = text.indexOf(word + " ");
38          while (index != -1) {
39              text.replace(index, index + word.length(), replacement);
40              count++;
41              index = text.indexOf(word + " ", index + replacement.length());
42          }
43          return count;
44      }
```

Figura 12: Metodo replace

- "getText()" es el método para obtener el texto almacenado en la Trie. Devuelve el texto almacenado en el StringBuilder text como una cadena de caracteres.

```
46      public String getText() {
47          return text.toString();
48      }
```

Figura 13: Metodo getText

- "TrieNode" es una clase interna que representa un nodo de la Trie. Tiene dos atributos: children, que es un mapa que mantiene las relaciones entre los caracteres y los nodos hijos, y isEndOfWord, un indicador que determina si este nodo es el final de una palabra. El constructor TrieNode() inicializa el mapa children como un nuevo HashMap y isEndOfWord como false.

```
50      private static class TrieNode {
51          private Map<Character, TrieNode> children;
52          private boolean isEndOfWord;
53
54          public TrieNode() {
55              children = new HashMap<>();
56              isEndOfWord = false;
57          }
58      }
59  }
```

Figura 14: Clase TrieNode

2.2. Pruebas

- **Insertar**

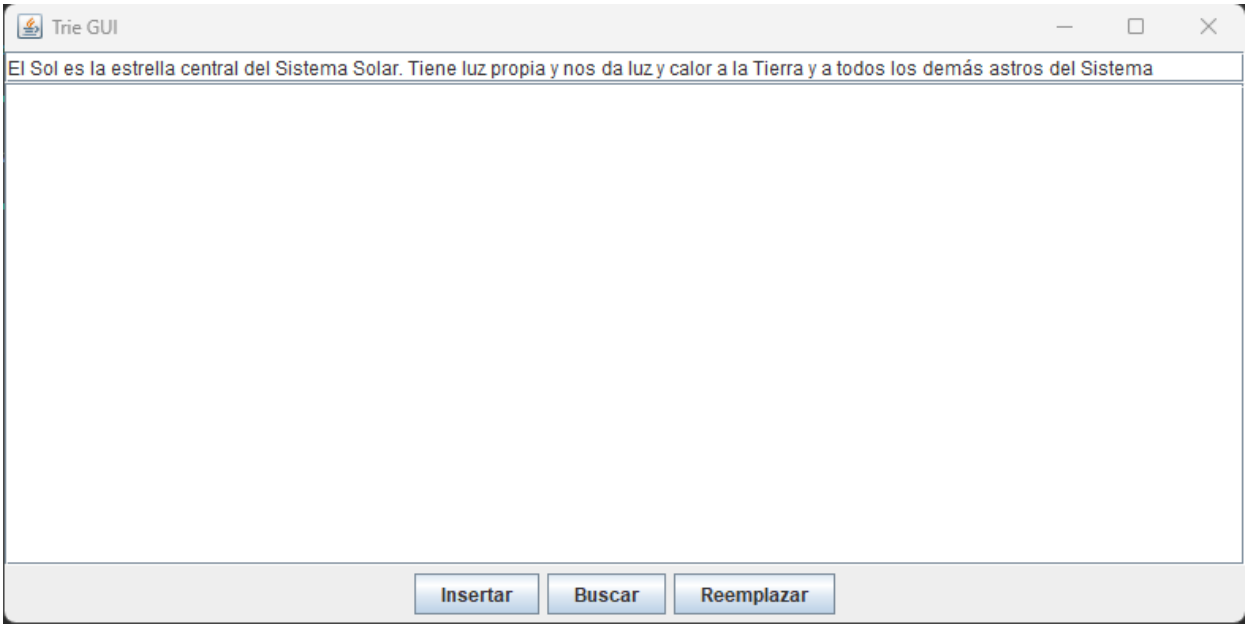

Figura 15: Ingresar texto

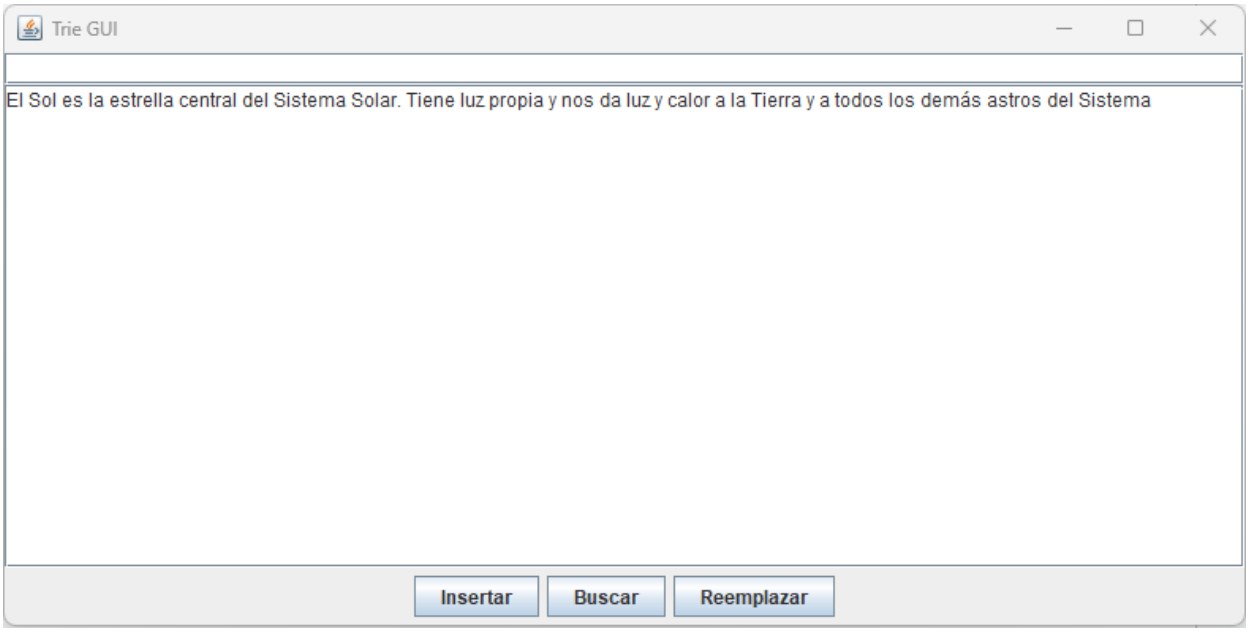

Figura 16: Mostrar texto

- **Buscar**

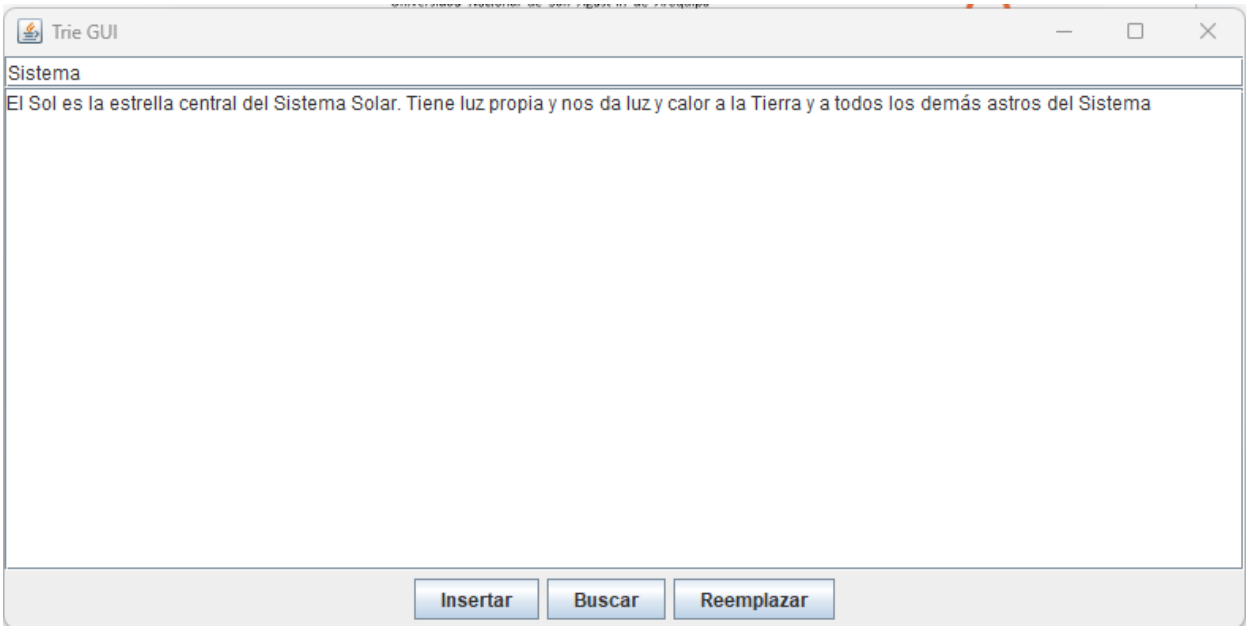

Figura 17: Ingresar palabra

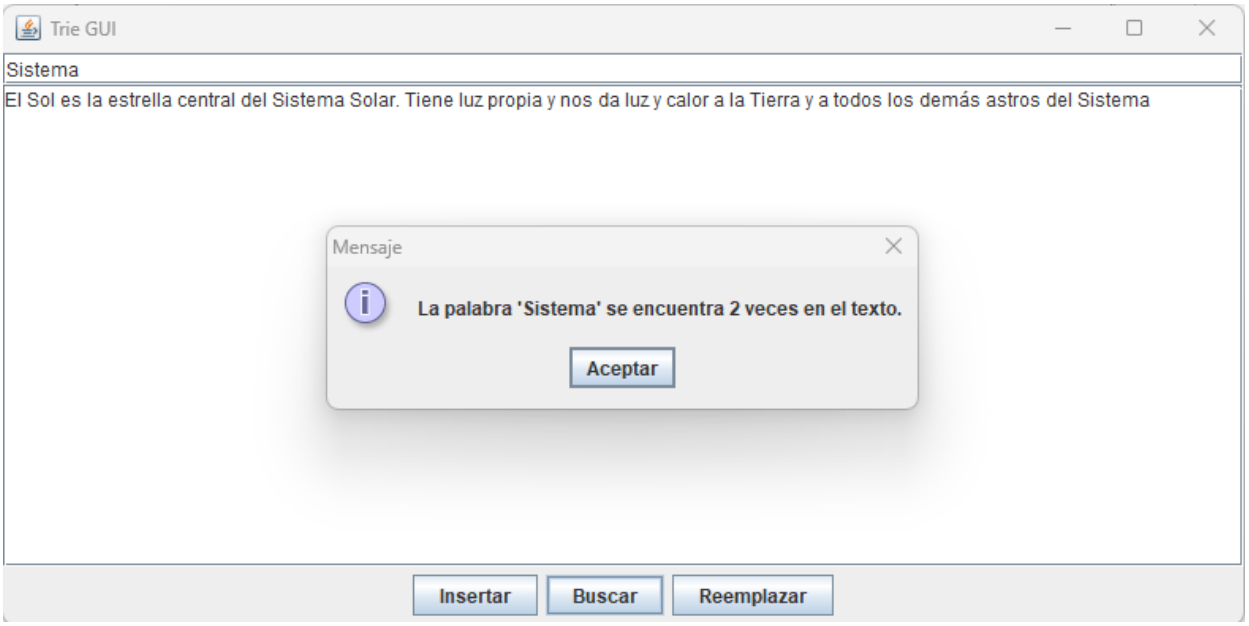

Figura 18: Mensaje de busqueda

- **Reemplazar**

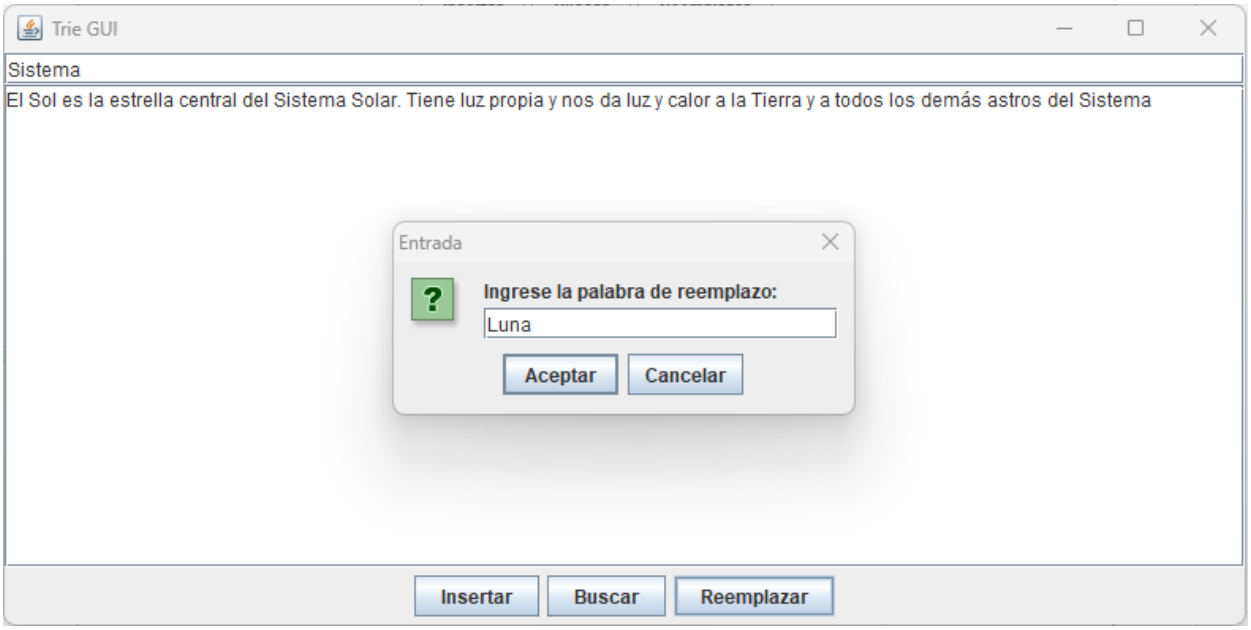

Figura 19: MPalabra a reemplazar

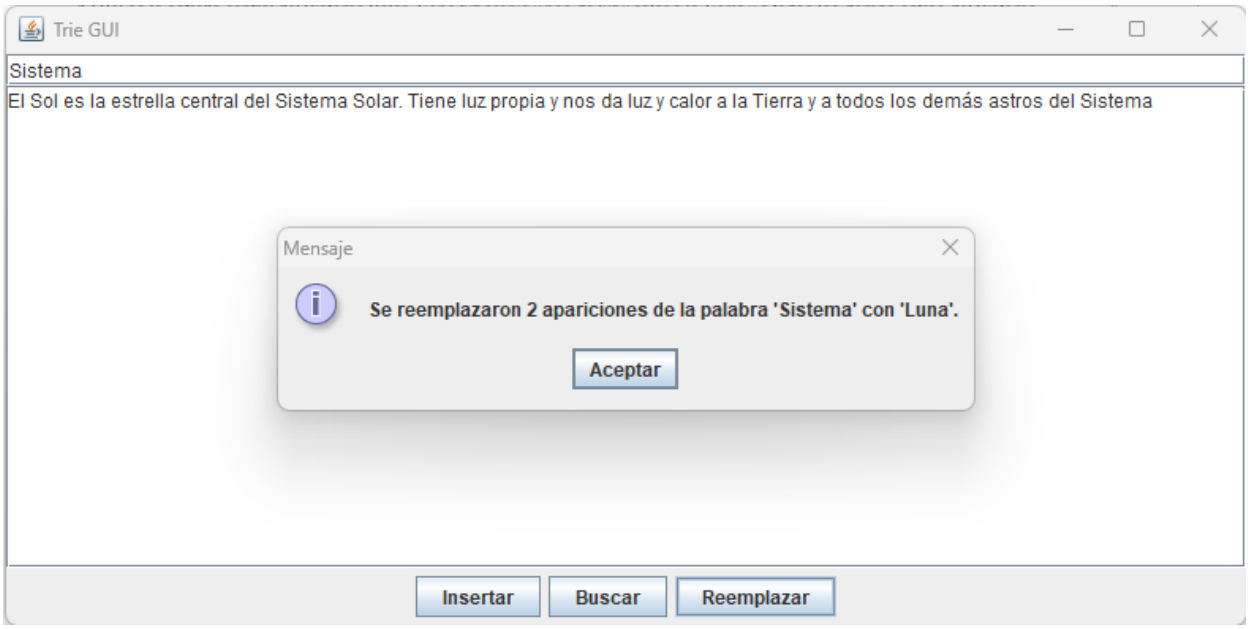

Figura 20: Mensaje de reemplazo

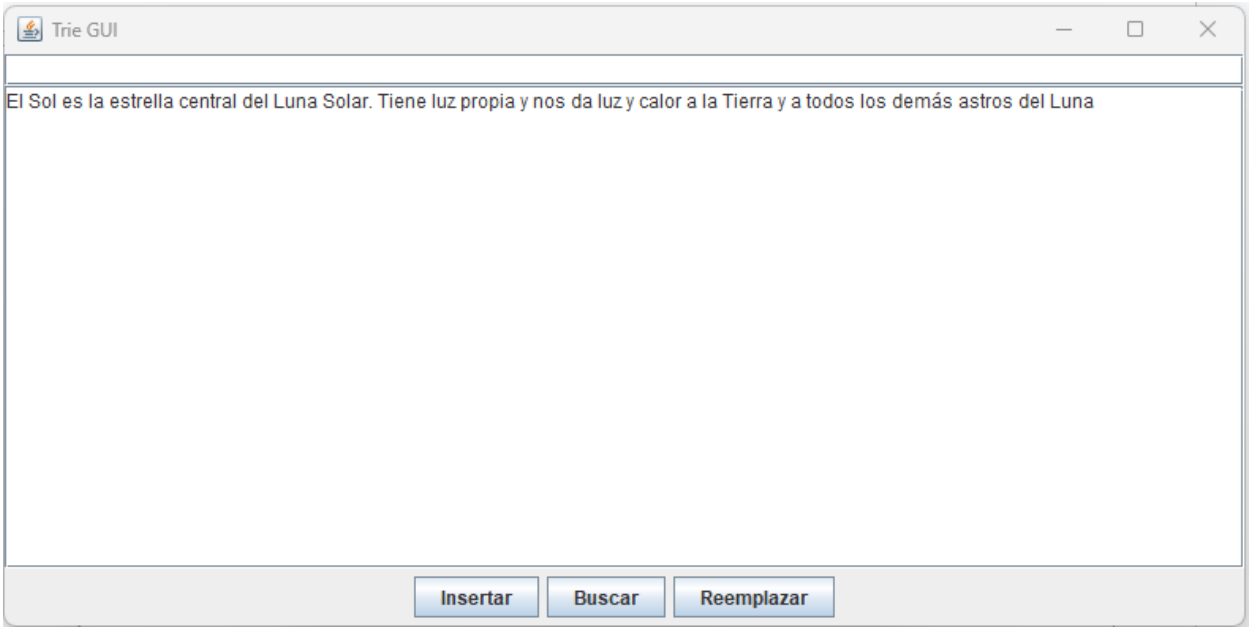

Figura 21: Resultado de reemplazo

2.3. Resolución de cuestionario

- **Explique. ¿Cómo se utiliza esta estructura de datos para almacenar prefijos?.**

La estructura de datos Trie es especialmente útil para almacenar y buscar palabras o cadenas de caracteres de manera eficiente. Se utiliza comúnmente en tareas que involucran búsqueda rápida de palabras o prefijos, como autocompletar en motores de búsqueda, correctores ortográficos y sistemas de búsqueda de palabras clave.

- **¿Cómo realizo la funcionalidad de reemplazar texto?**

Recibe dos parámetros: word es la palabra que se va a reemplazar y replacement es la palabra de reemplazo. Busca la primera aparición de word en el texto utilizando indexOf. Mientras encuentre word en el texto (indexOf devuelve un índice válido), reemplaza la palabra con replacement utilizando el método replace del StringBuilder. Incrementa count para mantener un registro de cuántos reemplazos se han realizado. Luego, busca la siguiente aparición de word a partir del índice siguiente al reemplazo realizado. El método devuelve la cantidad de reemplazos realizados.