

# Informe de Laboratorio 07

## Tema: Tries

Nota

Estudiantes	Escuela	Asignatura
Jorge Luis Pachari Quispe jpachari@unsa.edu.pe Nombre2 jparedesmes@unsa.edu.pe Nombre3 mtinta@unsa.edu.pe Nombre4 correo@unsa	Escuela Profesional de Ingeniería de Sistemas	Estructura de Datos y Algoritmos Semestre: III Código: 1702124

Laboratorio	Tema	Duración
07	Tries	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - A	Del 10 julio 2023	Al 17 julio 2023

Docente
Prof Edith Giovanna Cano Mamani

## 1. Soluciones y Resultados

- Aquí se encuentra el enlace del repositorio: <https://github.com>

### 1.1. Proyecto

- Clase base utilizada para el trie (trieNode)

Listing 1: TrieNode.java El cual es el nodo para nuestro trie

```
import java.util.HashMap;  
import java.util.Map;  
class TrieNode {  
    private Map<Character, TrieNode> children;  
    private boolean isWord;  
  
    public TrieNode() {  
        children = new HashMap<>();  
    }  
}
```

```
        isWord = false;
    }

    public Map<Character, TrieNode> getChildren() {
        return children;
    }

    public boolean isWord() {
        return isWord;
    }

    public void setWord(boolean word) {
        isWord = word;
    }
}
```

s

- Clase utilizada para el trie, la cual utiliza nodosTrie

Listing 2: Trie.java Aqui utilizamos una raiz y la indicamos en el constructor

```
import java.util.Map;

public class Trie {
    private TrieNode root;

    public Trie() {
        root = new TrieNode();
    }
}
```

- Metodo de insercion para el trie. Para insertar comparamos las veces hasta que no cumpla, alli empezara a crear nodos y al final colocara el correspondiente final de hoja.

Listing 3: Trie.java Metodo insert()

```
public void insert(String word) {
    TrieNode current = root;

    for (int i = 0; i < word.length(); i++) {
        char ch = word.charAt(i);
        Map<Character, TrieNode> children = current.getChildren();
        TrieNode node = children.get(ch);

        if (node == null) {
            node = new TrieNode();
            children.put(ch, node);
        }

        current = node;
    }

    current.setWord(true);
}
```

- Para la búsqueda se compara toda la clave, si no termina la clave, significara que no esta la clave. Al terminar la clave, debe verificar que si exista con el setWord

Listing 4: Trie.java Aqui se encuentra el metodo search()

```
public boolean search(String word) {
    TrieNode node = searchNode(word);
    return node != null && node.isWord();
}
private TrieNode searchNode(String word) {
    TrieNode current = root;

    for (int i = 0; i < word.length(); i++) {
        char ch = word.charAt(i);
        Map<Character, TrieNode> children = current.getChildren();
        TrieNode node = children.get(ch);

        if (node == null) {
            return null;
        }

        current = node;
    }

    return current;
}
```

- Para reemplazar una palabra primero debemos de buscarla, una vez verificada insertara la nueva palabra en su lugar. De esta manera, la palabra .oldWord.<sup>es</sup> reemplazada por "newWord.<sup>en</sup> la estructura de datos Trie.

Listing 5: Trie.java Aqui se encuentra el metodo replace()

```
public void replace(String oldWord, String newWord) {
    TrieNode node = searchNode(oldWord);

    if (node != null && node.isWord()) {
        node.setWord(false);
        insert(newWord);
    }
}
```

## 1.2. Resultados

- Para confirmar nuestro codigo por consola, aun sin la interfaz grafica, se creo una clase main para probar los metodos

Listing 6: Main.java Aqui se realizaron las pruebas de los metodos

```
public class Main {
    public static void main(String[] args) {
        Trie trie = new Trie();

        // Insertar palabras en el trie
        trie.insert("hola");
        trie.insert("adios");
    }
}
```

```

    trie.insert("hoy");
    trie.insert("ma(n)ana");

    // Buscar palabras en el trie
    System.out.println(trie.search("hola")); // true
    System.out.println(trie.search("bye")); // false

    // Reemplazar palabras en el trie
    trie.replace("hola", "hello");
    System.out.println(trie.search("hola")); // false
    System.out.println(trie.search("hello")); // true
}
}

```

■ Resultados por consola

```


jorgeluis@jorgeluis:~/Escritorio/edaLab$ cd /home/jorgeluis/Escritorio/edaLab ; /usr/bin/env /usr/lib/jvm/zulu-20-amd64/bin/java -agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:34631 --enable-preview -XX:+ShowCodeDetailsInExceptionMessages -cp /home/jorgeluis/.config/Code/User/workspaceStorage/8132d10693ca7a7282ee1d4b81051942/redhat.java/jdt_ws/edaLab_acldfff7/bin Main
true
false
false
true

```

■ Buscar



■ Insertar



The screenshot shows a software window with a title bar containing a small icon and standard window controls (minimize, maximize, close). Below the title bar are three tabs: 'Insertar' (selected), 'Buscar', and 'Reemplazar'. The main area of the window contains the text 'Ingrese Texto:' followed by a text input field containing 'Ingrese Texto'. In the bottom right corner, there is a button labeled 'Insertar'.

- Reemplazar



The screenshot shows the same software window, but now the 'Reemplazar' tab is selected. The main area contains two text input fields. The first is labeled 'Ingrese texto a buscar:' and contains 'Ingrese Texto'. The second is labeled 'Ingrese texto a reemplazar:' and also contains 'Ingrese Texto'. In the bottom right corner, there is a button labeled 'Buscar y Reemplazar'.

## 2. Conclusiones

- 

## 3. Referencias y Bibliografía

- Enlace del repositorio: <https://github.com/JorgeLuis1907/pw2-lab-d-23a/tree/main/Laboratorio5>