División de Ciencias de la Ingeniería
Centro Universitario de Occidente
Universidad de San Carlos de Guatemala

Enlace Repositorio
https://github.com/MunguiaMander/MyDBMS
Manual Tecnico
Proyecto Final Laboratorio Estructura de Datos.
24/05/2023

MARCO JOSE MUNGUIA ALVA - 201931804

MyDBMS

Instrucciones de instalación y compilación

1. Dirígete a la carpeta donde están los archivos del proyecto y dale permisos al script:

```
chmod +x InitProject.sh
```

2. Una vez compilado el programa, puedes ejecutar en la línea de comandos esta orden para que inicie el programa:

```
./InitProjecto.sh
```

Estructura XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation
=
"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xs
d/maven-4.0.0.xsd"
>
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.tlacuachesdevs.mydbms</groupId>
    <artifactId>MyDBMS</artifactId>
    <version>1.0-SNAPSHOT</version>
    <properties>
        <project.build.sourceEncoding>UTF-8</
project.build.sourceEncoding>
        <maven.compiler.source>17</maven.compiler.source>
        <maven.compiler.target>17</maven.compiler.target>
        <exec.mainClass>com.tlacuachesdevs.mydbms.MyDBMS</
exec.mainClass>
    </properties>
    <dependencies>
        <dependency>
            <groupId>guru.nidi</groupId>
            <artifactId>graphviz-java</artifactId>
            <version>0.17.0</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-simple</artifactId>
            <version>1.7.30</version>
<!-- use the version compatible with your project -->
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-assembly-plugin</artifactId>
                <version>3.3.0</version>
                <configuration>
                    <descriptorRefs>
                        <descriptorRef>jar-with-dependencies</
descriptorRef>
                    </descriptorRefs>
                    <archive>
                        <manifest>
                            <addClasspath>true</addClasspath>
                            <mainClass
>com.tlacuachesdevs.mydbms.MyDBMS</mainClass>
                        </manifest>
                    </archive>
                </configuration>
                <executions>
                    <execution>
                        <id>make-assembly</id>
                        <phase>package</phase>
                        <goals>
                            <goal>single</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>

</project>
```

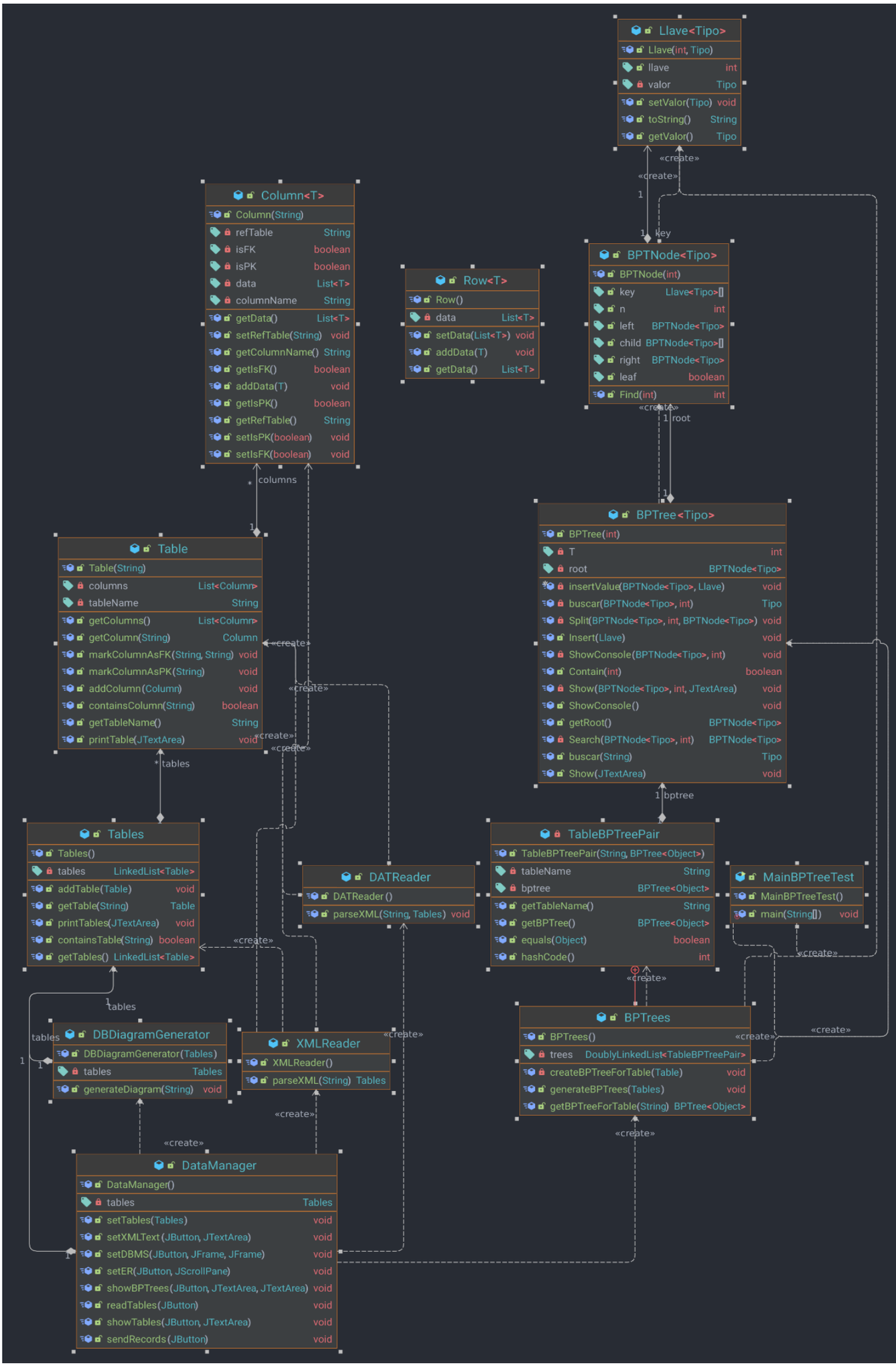
- **<groupId>, <artifactId>, y <version>:** Estos elementos forman la identidad del proyecto. En este caso, com.tlacuachesdevs.mydbms es el identificador único del grupo para el proyecto, MyDBMS es el identificador del proyecto, y 1.0-SNAPSHOT es la versión actual del proyecto.

- **<properties>:** Este es el lugar donde puedes definir varias propiedades del proyecto que se pueden reutilizar. Aquí se establece la codificación de los archivos fuente, la versión del compilador de Maven y la clase principal a ejecutar.

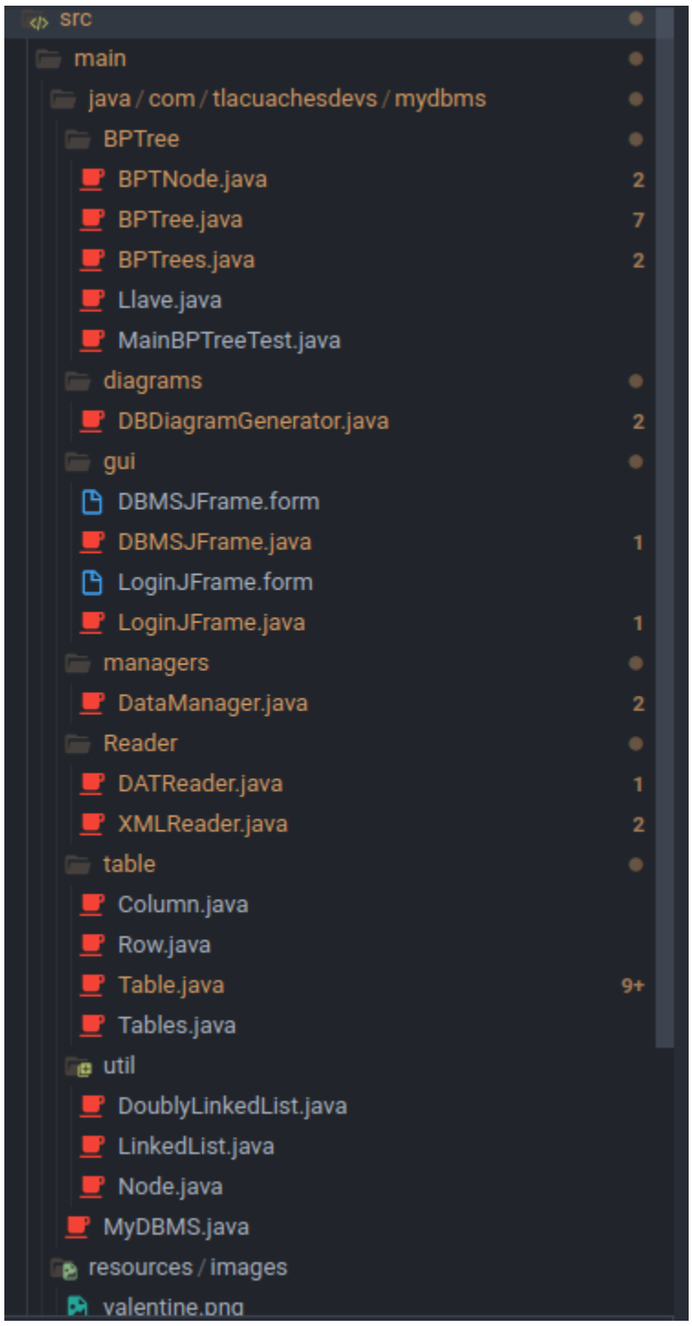
- **<dependencies>:** Aquí es donde se definen todas las dependencias que tu proyecto necesita. Hay dos dependencias definidas en este archivo:
  graphviz-java versión 0.17.0, es una librería de Java para la visualización de gráficos. Se utiliza para convertir descripciones de gráficos en un lenguaje de gráficos DOT en representaciones visuales.

- **<build>:** Esta sección se utiliza para configurar aspectos relacionados con la compilación del proyecto. Aquí está definido un plugin, maven-assembly-plugin, que se utiliza para crear un JAR con todas las dependencias necesarias. La configuración especifica que se debe incluir la ruta de clases en el archivo MANIFEST del JAR y define la clase principal que se debe ejecutar cuando se ejecuta el JAR.

Diagrama de Clases:



Estructura del Proyecto(Clases):



Arbol B+:

Clase BTNode<Tipo>

Clase genérica que representa el nodo del Arbol+

```java
package com.tlacuachesdevs.mydbms.BPTree;

/**
 *
 * @author mander
 */
public class BPTNode<Tipo> {

    public int n;
    public Llave<Tipo> key[];
    public BPTNode<Tipo> child[];
    public boolean leaf = true;

    public BPTNode<Tipo> left;
    public BPTNode<Tipo> right;

    public BPTNode(int T) {
        this.key = new Llave[2 * T - 1];
        this.child = new BPTNode[2 * T];
    }

    public int Find(int k) {
        for (int i = 0; i < this.n; i++) {
            if (this.key[i].llave == k) {
                return i;
            }
        }
        return -1;
    }
}
```

Clase Llave<Tipo>

Clase genérica que representa la llave de cada hoja del árbol b+

```java
package com.tlacuachesdevs.mydbms.BPTree;

/**
 *
 * @author mander
 */
public class Llave<Tipo> {

    public int llave;
    private Tipo valor;

    public Llave(int llave, Tipo valor) {
        this.llave = llave;
        this.valor = valor;
    }

    public Tipo getValor() {
        return valor;
    }

    public void setValor(Tipo valor) {
        this.valor = valor;
    }

    @Override
    public String toString() {
        return llave + ":" + valor;
    }
}
```

Clase BPTree<Tipo>

Esta clase contiene los métodos para que un árbol b+ funcione correctamente como dividirse, insertar, buscar:

```
private BPTNode<Tipo> Search(BPTNode<Tipo> x, int key) {
    int i = 0;
    if (x == null) {
        return x;
    }
    for (i = 0; i < x.n; i++) {
        if (key < x.key[i].llave) {
            break;
        }
        if (key == x.key[i].llave) {
            return x;
        }
    }
    if (x.leaf) {
        return null;
    } else {
        return Search(x.child[i], key);
    }
}
```

```
// Splitting the node
    private void Split(BPTNode<Tipo> x, int pos, BPTNode<Tipo> y) {
        BPTNode<Tipo> z = new BPTNode(T);
        z.leaf = y.leaf;
        z.n = T - 1;
        for (int j = 0; j < T - 1; j++) {
            z.key[j] = y.key[j + T];
        }
        if (!y.leaf) {
            for (int j = 0; j < T; j++) {
                z.child[j] = y.child[j + T];
            }
        }
        y.n = T - 1;
        for (int j = x.n; j >= pos + 1; j--) {
            x.child[j + 1] = x.child[j];
        }
        x.child[pos + 1] = z;

        for (int j = x.n - 1; j >= pos; j--) {
            x.key[j + 1] = x.key[j];
        }
        x.key[pos] = y.key[T - 1];
        x.n = x.n + 1;
    }
```

```
// Inserting a value
   public void Insert(final Llave key) {
       BPTNode<Tipo> r = root;
       if (r.n == 2 * T - 1) {
           BPTNode<Tipo> s = new BPTNode(T);
           root = s;
           s.leaf = false;
           s.n = 0;
           s.child[0] = r;
           Split(s, 0, r);
           insertValue(s, key);
       } else {
           insertValue(r, key);
       }
   }
```