UNIVERSIDAD SAN CARLOS DE GUATEMALA
CENTRO UNIVERSITARIO DE OCCIDENTE
DIVISIÓN CIENCIAS DE LA INGENIERÍA
ESTRUCTURA DE DATOS
ING. PEDRO LUIS DOMINGO
PRIMER SEMESTRE DE 2023

Proyecto II - Database Management System

por

Luis Alejandro Méndez Rivera, 202030627

Martes 11 de Abril, 2023

Database Management System (Desktop App)

Este manual contiene la documentación general sobre temas y aspectos relacionados al desarrollo de la Aplicación de Escritorio "Database Management System" solicitada para los siguientes criterios:

- Manejo de Tablas/Estructuras e Inserción de datos como Filas/Tuplas (ListasEnlazadas, NodosDeListas, Clases) en diversos contextos.
- Uso Arbol B+ (lo cual no se pudo concretar pero se optó por el uso de Listas Enlazadas y uso de Nodos)
- Archivos de Etiqueta XML.
- Punteros (referencias) hacia otros Objetos del mismo o diferente tipo.
- Captura de información y despliegue de la misma entre Ventanas.

Listas enlazadas

Una lista enlazada es una estructura de datos lineal en la que los elementos se almacenan en nodos, cada uno de los cuales contiene un valor y un puntero al siguiente nodo de la lista. El último nodo de la lista apunta a null. Esta estructura de datos es muy útil cuando se necesitan insertar o eliminar elementos con frecuencia, ya que no es necesario mover los elementos contiguos para hacer espacio o eliminar un elemento.

Las operaciones más comunes en una lista enlazada son:

- ☐ Insertar un elemento al principio o al final de la lista.
- ☐ Eliminar un elemento de la lista (por un Atributo de la Clase que alberga su nodo).
- ☐ Buscar un elemento en la lista (recorriendo la lista basado en un Atributo de la Clase que alberga cada nodo).
- ☐ Referencia/Movimiento a un nodo Anterior y/o Siguiente. Así como también puede ser Primero y/lo Último (dependiendo de la necesidad).

Funcionalidad

Requisitos

La aplicación Database Management System fue dearrollada en el lenguaje de programación Java 11, por lo cual, previo a poder ejecutar el programa, es necesario descargar e instalar el JDK de Java (o instalar Java SE Development Kit, que incluye el JDK):

Java SE: https://www.java.com/es/download/

Opcional: Puede descargar NetBeans con entorno para Java (para correr el JAR y ver las impresiones de Consola)

Luego de contar con el requisito indispensable que permitirá el análisis de código .class (compilado de Java), podremos descargar el repositorio que contiene el codigo fuente y el Archivo ".jar" que es el programa final en este enlace: https://github.com/MenWic/DBManagementeService-EDD

Una vez descargados los recursos, podremos correr el programa dando doble click al Archivo de formato "**JAR**" ubicado en:
[***RutaDelRepositorio***] "\DBMagementSystem\target"

También podemos correr el programa mediante el siguiente comando:

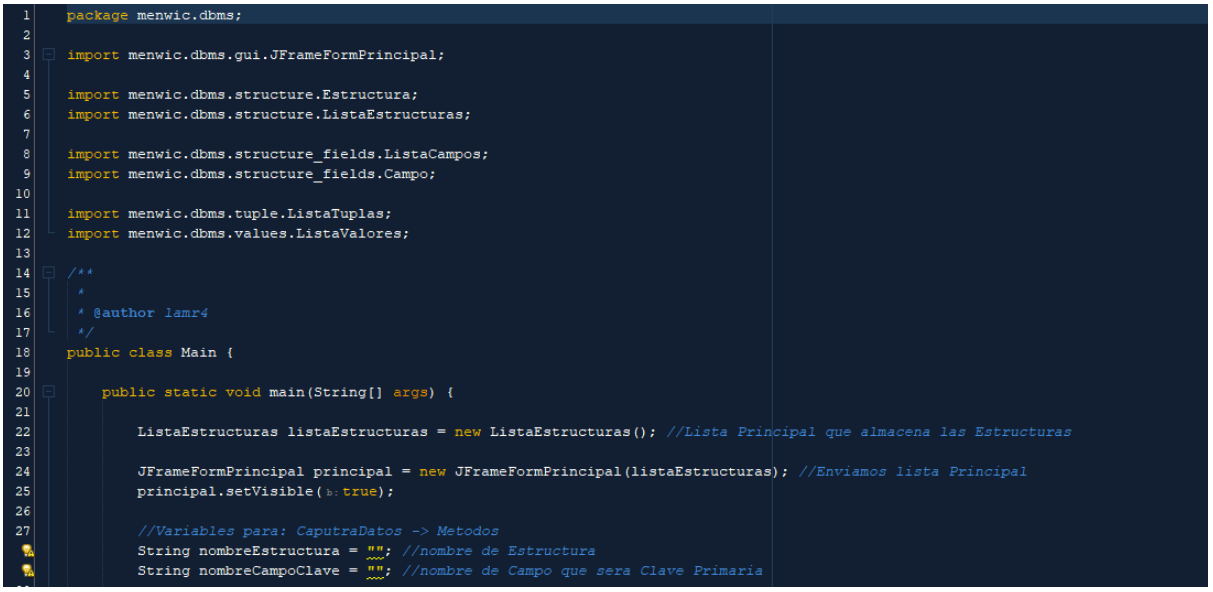
```
1. cd [RutaDeRepositorio]/DBManagementSystem/target
2. java -jar dbms-1.0-SNAPSHOT.jar
```

Código Fuente del Programa

Main.java

Metodo principal que contiene la “listaEstructura” de tipo ListaEnlazada inicial que sera la lista enlazada que almacenara las estructuras/tablas que se instancien/lean/creen.

Este metodo ademas que invoca una Instancia de JFramePrincipal, el cual es una ventana visual de Java, la cual contendra las variables globales iniciales para el funcionamiento y manipulacion de datos y creacion/instancia de diversos objetos



```java
package menwic.dbms;

import menwic.dbms.gui.JFrameFormPrincipal;

import menwic.dbms.structure.Estructura;
import menwic.dbms.structure.ListaEstructuras;

import menwic.dbms.structure_fields.ListaCampos;
import menwic.dbms.structure_fields.Campo;

import menwic.dbms.tuple.ListaTuplas;
import menwic.dbms.values.ListaValores;

/**
 *
 * @author lamr4
 */
public class Main {

    public static void main(String[] args) {

        ListaEstructuras listaEstructuras = new ListaEstructuras(); //Lista Principal que almacena las Estructuras

        JFrameFormPrincipal principal = new JFrameFormPrincipal(listaEstructuras); //Enviamos lista Principal
        principal.setVisible(b: true);

        //Variables para: CaputraDatos -> Metodos
        String nombreEstructura = ""; //nombre de Estructura
        String nombreCampoClave = ""; //nombre de Campo que sera Clave Primaria
```

ListaEsructuras.java

Esta clase es en esencial, ya que es la forma en que funciona el gestor de DB, ya que es una lista enlazada compuesta por objetos “NodoEstructura”, los cuales ademas de tener apuntadores a otros NodoEstructura, alberga al objeto principal Estructura que veremos mas adelante.

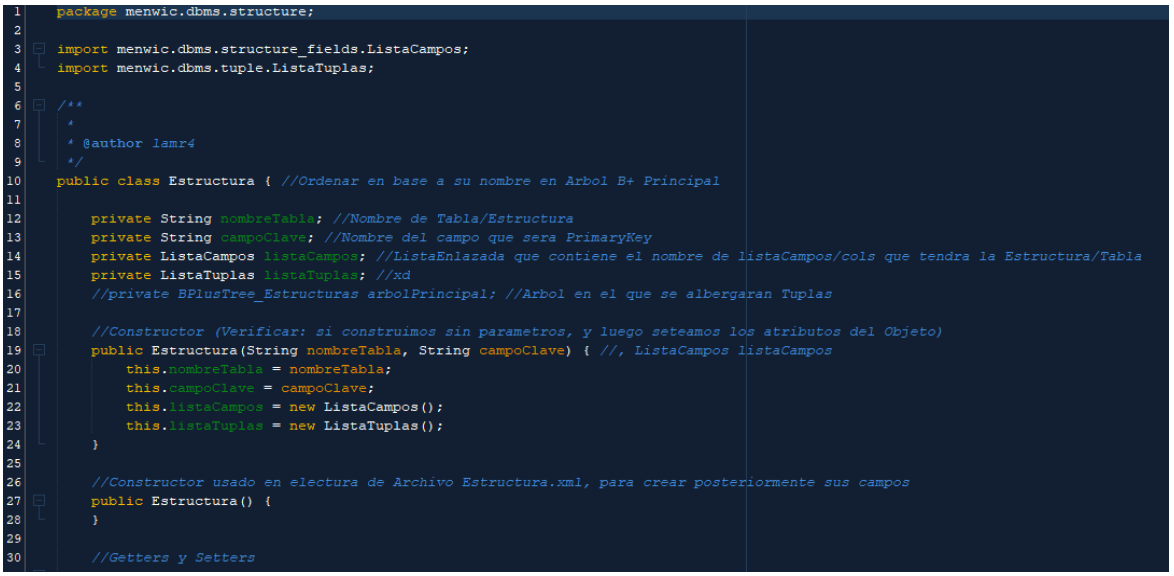
Presenta metodos para:

- insertarFinal(nodo a insertar), eliminarFInal()
- getNodoPorNombreEstructura(nombre de la estructura que busca)
- encontroEstructura(nombre de Estructura que busca)

```java
package menwic.dbms.structure;

/**
 *
 * @author lamr4
 */
public class ListaEstructuras {

    //Atributos de cada Instancia ListaEstructuras
    private NodoEstructura primero;
    private NodoEstructura ultimo;
    private int size = 0; //Contador de nodos

    //Constructor
    public ListaEstructuras() {
        this.primero = null;
        this.ultimo = null;
    }

    //Getters y Setters
    public int getSize() {
        return size;
    }

    public void setSize(int size) {
        this.size = size;
    }

    public NodoEstructura getPrimero() {
        return primero;
```

Estructura.java

Clase que instancia una estructura de datos “Estructura/Tabla” la cual presenta un nombre, un campo clave primaria y una lista de campos (columnas de la tabla). Además presenta una lista de tuplas, las cuales son inserciones de datos que contienen valores asociados a un campo de la estructura. Presenta además metodos de acceso y actualizacion para cada Atributo de la clase.



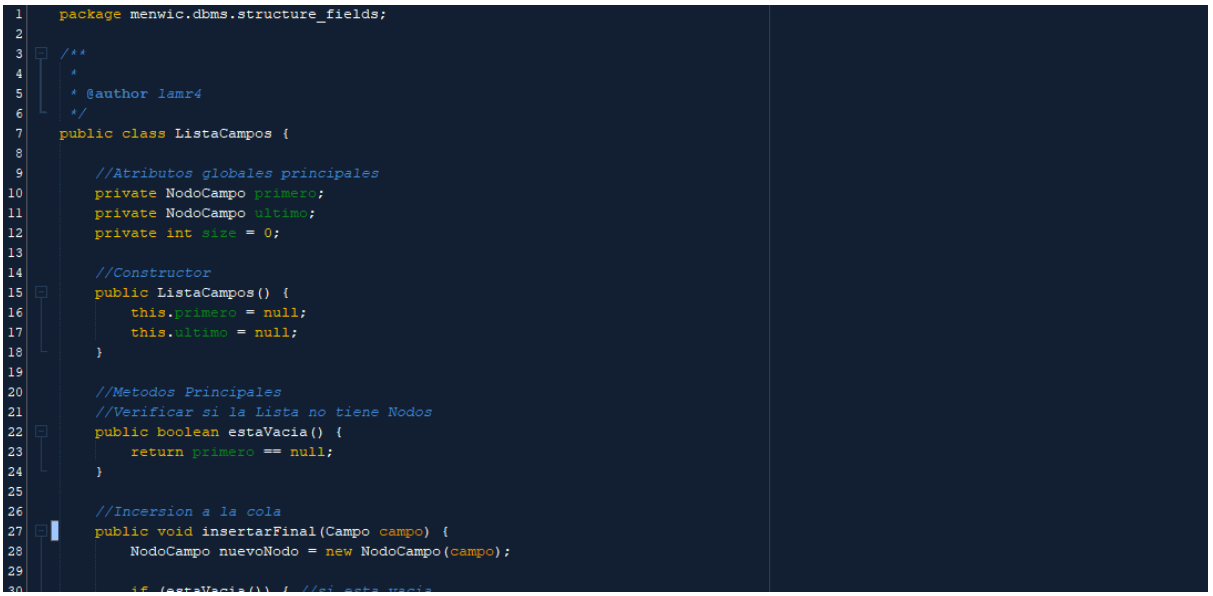
```java
package menwic.dbms.structure;

import menwic.dbms.structure_fields.ListaCampos;
import menwic.dbms.tuple.ListaTuplas;

/**
 *
 * @author lamr4
 */
public class Estructura { //Ordenar en base a su nombre en Arbol B+ Principal

    private String nombreTabla; //Nombre de Tabla/Estructura
    private String campoClave; //Nombre del campo que sera PrimaryKey
    private ListaCampos listaCampos; //ListaEnlazada que contiene el nombre de listaCampos/cols que tendra la Estructura/Tabla
    private ListaTuplas listaTuplas; //xd
    //private BPlusTree_Estructuras arbolPrincipal; //Arbol en el que se albergaran Tuplas

    //Constructor (Verificar: si construimos sin parametros, y luego seteamos los atributos del Objeto)
    public Estructura(String nombreTabla, String campoClave) { //, ListaCampos listaCampos
        this.nombreTabla = nombreTabla;
        this.campoClave = campoClave;
        this.listaCampos = new ListaCampos();
        this.listaTuplas = new ListaTuplas();
    }

    //Constructor usado en electura de Archivo Estructura.xml, para crear posteriormente sus campos
    public Estructura() {
    }

    //Getters y Setters
```

ListaCampos.java

Esta clase se utiliza para guardar las columnas/campos de una tabla y alberga nodos los cuales además de presentar referencias a otros nodos, contienen los objetos de tipo “Campo”, los cuales presentan el nombre del campo, el tipo de entero que deberá presentar cada tupla que se asocie a este, etc.

Presenta metodos para:

- insertarFinal(nodo a insertar), eliminarFInal()
- getNodoCampoPorNombreCampo(nombre de la estructura que busca)
- returnCampos()



```java
package menwic.dbms.structure_fields;

/**
 *
 * @author lamr4
 */
public class ListaCampos {

    //Atributos globales principales
    private NodoCampo primero;
    private NodoCampo ultimo;
    private int size = 0;

    //Constructor
    public ListaCampos() {
        this.primero = null;
        this.ultimo = null;
    }

    //Metodos Principales
    //Verificar si la Lista no tiene Nodos
    public boolean estaVacia() {
        return primero == null;
    }

    //Incersion a la cola
    public void insertarFinal(Campo campo) {
        NodoCampo nuevoNodo = new NodoCampo(campo);

        if (estaVacia()) { //si esta vacia
```

Campo.java

Clase que instancia una estructura de datos “Campo” que albergan las características que debe tener cada campo/columna perteneciente a una tabla. Tiene la característica opcional de tener una referencia/apuntador hacia el campo de otra Estructura, simulando así una “llave foránea”. Presenta además métodos de acceso y actualización para cada Atributo de la clase.

```java
package menwic.dbms.structure_fields;

/**
 *
 * @author lamr4
 */
public class Campo {

    //Atributos globales principales
    private String nombreCampo;
    private String tipoDato;
    private Campo campoReferencia; //Puntero OPCIONAL hacia otro Campo (de otra Tabla/Estructura)

    //Constructor: Campos sin campoReferencia a otra Tabla/Entidad
    public Campo(String nombreCampo, String tipoDato) {
        this.nombreCampo = nombreCampo;
        this.tipoDato = tipoDato;
    }

    // Getters y Setters
    public String getNombreCampo() {
        return nombreCampo;
    }

    public void setNombreCampo(String nombreCampo) {
        this.nombreCampo = nombreCampo;
    }

    public String getTipoDato() {
        return tipoDato;
```

ListaValores.java

Esta clase se utiliza para guardar “NodoValor”, los cuales albergan un valor que se utiliza para representar una tupla y se asocia a un Campo de una Estructura. Y presenta apuntadores a nodos adyacentes en la lista.
Presenta metodos para:
- insertarFinal(nodo a insertar), eliminarFInal()
- getNodoValorPorNombreCampo(nombre del campo que busca)
- actualizarValor(nombreCampo asociado, nuevoValor), getValores()

```java
package menwic.dbms.values;

/**
 *
 * @author lamr4
 */
public class ListaValores {
    //Variables globales
    private NodoValor primero;
    private NodoValor ultimo;
    private int size;

    //COnstructor
    public ListaValores() {
        this.primero = null;
        this.ultimo = null;
    }

    //METODOS PRINCIPALES
    //Verificar si la Lista no tiene Nodos
    public boolean estaVacia() {
        return primero == null;
    }

    //Incersion a la cola
    public void agregarAlFinal(Valor valor) {
        NodoValor nuevoNodo = new NodoValor(valor);

        if (estaVacia()) { //si esta vacia
            primero = nuevoNodo;
```

Valor.java

Clase que instancia una estructura de datos “Valor” la cual representa un dato perteneciente a una fila/tupla, relacionado mediante la referencia hacia a una columna/campo de una tabla. Sus atributos son “contenido” el cual es el valor perteneciente a una tupla y “nombreCampo” el cual es un texto para buscar un campo de la estructura a la que pertenece el valor y si se encuentra, relacionarlo.

```java
package menwic.dbms.values;

/**
 *
 * @author lamr4
 */
public class Valor {

    //Variables globales
    public String contenido;
    public String nombreCampo;

    //Constructor
    public Valor(String valor, String nombreCampo) {
        this.contenido = valor;
        this.nombreCampo = nombreCampo;
    }

    //Getters y Setters
    public String getContenido() {
        return contenido;
    }

    public void setContenido(String contenido) {
        this.contenido = contenido;
    }

    public String getNombreCampo() {
        return nombreCampo;
    }
```

ListaTuplas.java

Esta clase instancia filas de datos, cuyos datos/valores, deben asociarse a un campo de la estructura/tabla a la cual pertenecen todos los mencionados.
Presenta métodos para:
- insertarFinal(nodo a insertar), eliminarFInal()
- getNodoTuplaPorNombreClave(campo clave de estructura)
- returnTuplas()

```java
package menwic.dbms.tuple;

/**
 *
 * @author lamr4
 */
public class ListaTuplas {

    //Variables globales
    private NodoTupla primero;
    private NodoTupla ultimo;
    private int size;

    //Constructor
    public ListaTuplas() {
        this.primero = null;
        this.ultimo = null;
    }

    //Verificar si la Lista no tiene Nodos
    public boolean estaVacia() {
        return primero == null;
    }

    //Incersion a la cola
    public void agregarAlFinal(Tupla tupla) {
        NodoTupla nuevoNodo = new NodoTupla( valor: tupla);

        if (estaVacia()) { //si esta vacia
            primero = nuevoNodo;
```

Tupla.java
Esta clase es una estructura que almacena una “listavalores”, los cuales significan los datos/valores de una tupla, las cuales que se instancian mediante la clase CreadorTuplas.

```java
package menwic.dbms.tuple;

import menwic.dbms.values.ListaValores;
import menwic.dbms.values.Valor;

/**
 *
 * @author lamr4
 */
public class Tupla {

    //public String nombreCampo;
    public String clave;
    public ListaValores listaValores;

    //Constructor
    public Tupla(ListaValores listaValores, String clave) {
        this.listaValores = listaValores;
        this.clave = clave;
    }

    //Metodo que crea una lista de valores a partir d eun array de valores
    public Tupla(Valor[] valores, String clave) {
        this.listaValores = new ListaValores();

        for (Valor itemValor : valores) { //Recorre cada nodo de valores
            this.listaValores.agregarAlFinal( valor: itemValor);
        }
        this.clave = clave; //retorna la clave de cada nodo en cada iteracion
    }
```

CreadorTupla.java
Es la clase que contiene múltiples métodos para crear, eliminar, settear y demas opciones y metodos de busqueda y comprobación necesarios para la manipulación de informacion

FileHandler.java
Clase encargada de seleccionar archivos de entrada y tambien manejar la captura de datos (es un Analizador) y dependiendo el tipo de Archivo de entrada, se iniciara uno u otro analizador de esta clase