MANUAL TECNICO

Repositorio en GitHub: https://github.com/x30061/josq-db

VERSION DE JAVA UTILIZADA: java 17.0.4.1 2022-08-18 LTS

VERSION DE MAVEN UTILIZADA: Apache Maven 3.8.2

LIBRERIAS UTILIZADAS:

- jdom2 2.0.6.1: indispensable para leer los documentos xml e interactuar masivamente con la base de datos.
- javafx 19: indispensable para que el usuario final interactue de forma amigable con la base de datos.

REQUISITOS DE USO: se necesita tener instalado la JRE y Graphviz.

A continuacion se muestra el diagrama de clases que representa la base de datos. Solo se muestran las clases indispensables para el funcionamiento de la base de datos.

Interactuar con la API:

La clase Interacciones ofrece los metodos para realizar operaciones en la base de datos, entre los cuales estan:

addColumna(tabla: String,columna: String,tipo: String)

addLlave(tabla: String,tipo: String)

addTabla(tabla: String)

deleteDato(tabla: String,llave: String,columna: String)

deleteLlave(tabla: String,llave: String)

setDato(tabla: String,llave: String,columna: String)

setRelacion(tabla: String,columna: String,relacion: String)

setLlavePrimaria(tabla: String,columna: String)

ARQUITECTURA DE LA BASE DE DATOS

MiBD + MiBD(nombre: String) - tablas: ArbolBp<String,Tabla> - nombre: String + addTabla(tabla: String): void + setLlavePrimaria(tabla: String,columna: String): void + setRelacion(tabla: String.columna: String, relacion: String): void + addColumna(tabla: String,columna: String,tipo: String): void + addLlave(tabla: String,llave: String): void + setDato(tabla: String,llave: String,columna: String,valor: String): void + deleteDato(tabla: String,llave: String,columna: String): void + deleteLlave(tabla: String,llave: String): void + getCadenaGraphviz(): StringBuilder Tabla Tabla(nombre: String) - nombre: String Columna - columnas: ArbolBp<String,Columna> # Columna(nombre: String,tipo: TiposAutorizados) - llavePrimaria: Columna datos: ArbolBp<String,String> - relaciones: ArbolBp<String,Columna> - tipo: TiposAutorizados - nombre: String # addColumna(columna: String,tipo: String): void # addLlave(llave: String): void # deleteDato(llave: String): void # deleteLlave(llave: String): void # setDato(llave: String,valor: String): void # deleteTupla(llave: String): void # getDato(llave: String): void # deleteValor(llave: String, columna: String): void # getCadenaGraphviz(): StringBuilder # setDato(llave: String,columna: String,valor: String): void ArbolBp<_Llave_ extends Comparable<_Llave_>, _Valor_> Nodo <_Llave_ extends Comparable<_Llave_>> - raiz: Nodo<_Llave_> # Nodo() - setEntrada(llave: _Llave_,valor: _Valor_): void - llaves: Object∏ + deleteEntrada(llave: _Llave_): void - numerDeLlaves: int - getHoja(llave: _Llave_): NodoHoja<_Llave_,_Valor_> - ancestro: Nodo <_Llave_> + getValor(llave: _Llave_): _Valor_ - izquierdo: Nodo <_Llave_> - derecho: Nodo <_Llave_> NodoHoja <_Llave_ extends Comparable<_Llave_>,_Valor_> + dealCarencia(): Nodo<_Llave_> + NodoHoja() + dealExceso(): Nodo<_Llave_> # ORDEN: int + getLlave(indice: int): _Llave_ - valores: Object∏ + setLlave(indice: int,llave: _Llave_): void + getNumeroDeLlaves(): int - deleteEnPosicion(indice: int): void + getPosicionDeLlave(llave: _Llave_): int + getPosicionDeLlave(llave: _Llave_): int + hasCarencia(): boolean + getValor(indice: int): _Valor_ + hasExceso(): boolean + setValor(indice: int,valor: _Valor_): void - addEntrada(indice: int,llave: _Llave_,valor: _Valor_): void + addLlave(llave: _Llave_, valor: _Valor_): void NodoRama<_Llave_ extends Comparable<_Llave_>> + NodoRama() # ORDEN: int # hijos: Object[]

- deleteEn(indice: int): void
- + getHijo(indice: int): Nodo<_Llave_>
- + getPosicionDeLlave(llave: _Llave_): int
- addEn(indice: int,llave: _Llave_,hijoIzq: Nodo<_Llave_>,hijoDer: Nodo<_Llave_>): void
- + setHijo(indice: int, hijo: Nodo<_Llave_>): void