Manual Técnico

Cristian Fernando Hernández Tello
202010905

Objetivos

General:

Brindar al lector una guía que contenga la información del manejo de clases, atributos, métodos y del desarrollo de la aplicación desarrollada en JavaScript para facilitar futuras actualizaciones y modificaciones realizadas por terceros.

Específicos:

- Dar más información al lector de las herramientas utilizadas para el desarrollo de la app como lo puede ser el editor, SO, entre otros.

- Proporcionar al lector una idea más precisa de los métodos y clases creadas para el desarrollo de la aplicación.

Introducción

Este manual técnico tiene como finalidad dar a conocer al lector que pueda requerir hacer modificaciones futuras al software el desarrollo de la aplicación indicando el IDE utilizado para su creación, su versión, requerimientos del sistema, etc.

La aplicación tiene como objetivo el manejo de información y archivos Json para cargar masivamente, libros, usuarios y autores, para la interacción con compras de libros, y disponibilidad de los mismos.

Se generan reportes graphviz para mostrar gráficamente las estructuras de datos implementadas.

Solución

Para poder desarrollar esta práctica se analizó lo que se solicitaba y se implementó estructuras de datos básicas como listas simples, doble enlazadas, matrices, y árboles.

- Para la cargas masivas de datos, se utilizaron funciones propias de javascript que permiten trabajar con archivos json, luego de descomponerlas se ingresaron los datos a las estructuras correspondientes.
- Para el ordenamientos se utilizó los algoritmos de burbuja y quicksort pero con la complejidad de que fueron adaptados a una lista con nodos.
- Para la graficación se recorrieron las matrices y se crean los elementos dinámicamente según el tamaño de las mismas

Descripción de funciones

mostrarMatriz():

En esta función agrega en una variable el código dot que se utilizará para renderizarlo y luego mostrar en el html, va recorriendo la matriz por sus cabeceras y accediendo a los nodos

```
class MatrizOrtogonal{
    constructor(){
        this.listahorizontal = new ListaEncabezado();
    }

    llenarmatrizortogonal(){
        for (let index = 1; index < 26; index++) {
            this.listahorizontal.insertarlista("", "", "", "", "", "", "","",index)
        }
    }
    mostrarmatriz(){
        var posx = 1
        var dotMatriz='digraph L{\n node[shape=box fillcolor="#FFEDBB" style=filled]\n subgraph cluster_p{ \n label ="Libros de Fantasia"\n bgc
        var dotMatriz2='digraph L{\n node[shape=box3d fillcolor="none" style=filled]\n bgcolor="none"\n subgraph cluster_p{ \n label ="Libros d
        let uniones=""
        let uniones2=""
        var cabecerax = this.listahorizontal.buscarlista(posx)
        while(cabecerax != null){
            let alineacion="{rank=same;"
            // console.log("*************** x="+posx+"*****************")
            var numy = 1
            var tempy = cabecerax.abajo
            while(tempy != null){
                // console.log(tempy)
                dotMatriz+='nodo'+tempy.x+'_'+tempy.y+'[label="'+tempy.nombreLibro+'",fillcolor=white,group=0] \n'
                dotMatriz2+='nodo'+tempy.x+'_'+tempy.y+'[label="'+tempy.nombreLibro+'",fillcolor=yellow,group=0] \n'
                // console.log(tempy.nombreLibro+"("+tempy.x+","+tempy.y+")")
                if (tempy.abajo!=null) {
                    let auxUnion=tempy.abajo
                    uniones+='nodo'+tempy.x+'_'+tempy.y+'->'+'nodo'+auxUnion.x+'_'+auxUnion.y+'[dir=both color="black"] \n'
                    uniones2+='nodo'+tempy.x+'_'+tempy.y+'->'+'nodo'+auxUnion.x+'_'+auxUnion.y+'[dir=both color="none"] \n'
                }
                if (tempy.abajo==null) {
                    alineacion+='nodo'+tempy.x+'_'+tempy.y
                }else{
```

ordenarTop():

Para el ordenamiento del top de usuarios con más compras, se utilizó el algoritmo de burbuja adaptado a una lista doblemente enlazada, esta entra en un ciclo hasta que su apuntador siguiente sea nulo y va comprando si la cantidad del nodo siguiente es menor que la actual, para realizar el cambio.

```
ordenarTop(){
    if (this.size>1) {
        while (true) {
            let actual=this.inicio;
            let x=null;
            let y=this.inicio.siguiente;
            let cambio=false
            while (y!=null) {
                if (actual.cantidadLibros<y.cantidadLibros) {
                    cambio=true;
                    if (x!=null) {
                        let tmp=y.siguiente;
                        x.siguiente=y;
                        y.siguiente=actual;
                        actual.siguiente=tmp
                    }else{
                        let tmp2=y.siguiente;
                        this.inicio=y;
                        y.siguiente=actual;
                        actual.siguiente=tmp2
                    }
                    x=y;
                    y=actual.siguiente;
                }else{
                    x=actual;
                    actual=y;
                    y=y.siguiente
                }
            }if (!cambio) {
                break;
            }
        }
    }
}
```

insertar():

Para la inserción en la matriz dispersa, se recorre las listas de la cabeceras de estas, para crear el nodo que se va ingresar, solicita la posición en la cual desea ser ingresada y luego de ubicarse se ingresan sus datos y se conectan sus apuntadores con los nodos de arriba,abajo,anterior y siguiente, por si llega a ingresar un nodo en la misma fila o columan para enlazarlo ya sea a este nodo o al nodo cabecera.

```
1081    insertar(posx,posy,isbn, nombreAutor, nombreLibro, cantidad, fila, columna, paginas, categoria){
1082        let nuevaCelda=new NodoMatriz(isbn,nombreAutor,nombreLibro,cantidad,fila,columna,paginas,categoria,posx,posy);
1083        this.inicio=nuevaCelda;
1084
1085        //buscar si existen las cabeceras en la matriz;
1086        let nodoX=this.filas.getCabecera(posx);
1087        let nodoY=this.columnas.getCabecera(posy);
1088
1089        //comprobamos que la cabecera fila posX exista
1090        if (nodoX==null) {
1091            nodoX=new NodoCabezera(posx); //si nodox es nulo quiere decir que no existe cabecera fila posx
1092            this.filas.insertarNodoCabecera(nodoX);
1093        }
1094        if (nodoY==null) {
1095            nodoY=new NodoCabezera(posy);
1096            //si nodo_Y es nulo, quiere decir que no existe  columna pos_y
1097            this.columnas.insertarNodoCabecera(nodoY);
1098        }
1099        // insertar nueva celda en fila
1100        if (nodoX.getAcceso()==null) {
1101            nodoX.setAcceso(nuevaCelda)
1102        }else{
1103            if (parseInt(nuevaCelda.coordenadaY)<parseInt(nodoX.getAcceso().coordenadaY)) {
1104                nuevaCelda.setDerecha(nodoX.getAcceso());
1105                nodoX.getAcceso().setIzquierda(nuevaCelda);
1106                nodoX.setAcceso(nuevaCelda);
1107            }else{
1108                //de no cumplirse debemos movernos de izquierda a derecha buscando donde posicionar el nueva_celda nodoInterno
1109                let tmp=nodoX.getAcceso();
1110                console.log(tmp)
1111                while (tmp!=null) {
1112                    if (parseInt(nuevaCelda.coordenadaY)<parseInt(tmp.coordenadaY)) {
1113                        nuevaCelda.setDerecha(tmp);
1114                        nuevaCelda.setIzquierda(tmp.getIzquierda());
1115                        tmp.getIzquierda().setDerecha(nuevaCelda);
```