Manual Técnico

Kelly Mischel Herrera Espino
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Introducción:

En el presente manual se describen los métodos, funciones y clases en las cuales se encuentran. Cabe mencionar que se utilizaron varias estructuras para almacenar los distintos tipos de datos. También se utilizo el lenguajes JavaScript para poder cumplir con las expectativas del proyecto.

Archivos

app.js

ArbolBinario.js

CargarArchivo.js

ColaEspera.js

ListaAscendente.js

ListaCircular.js

MatrizOrtogonal.js

Archivo app.js

Clase app

Listado de métodos:

- **admin():** en este método se hacen las validaciones para poder dar acceso a los usuarios y a los administradores del programa. Se verifica que si el usuario es de tipo admin pueda ingresar a correctamente a la pagina del administrados, la cual contiene diferentes funcionalidades, si el usuario es de tipo “usuario” se da acceso a las páginas en las cuales este pueda agregar libros a su biblioteca pero no podrá cargar archivos de entrada.
- **cargarUsuarios():** se muestra la div con el id cargar, esto para poder mostrar en el html la etiqueta input de tipo file y así poder cargar el archivo de usuarios.
- **cargarAutores:** se muestra la div con el id cargar_autores, esto para poder mostrar en el html la etiqueta input de tipo file y así poder cargar el archivo de autores.
- **cargarLibros:** se muestra la div con el id cargar_libros, esto para poder mostrar en el html la etiqueta input de tipo file y así poder cargar el archivo de libros.

- **mostrarPag_principal():** Se cambia el display de los divs que corresponde a la pagina principal cuando un usuario no a iniciado sesión y se ocultan los divs que no corresponden a esa página.
- **mostrarPag_principalAdmin:** Se cambia el display de los divs que corresponde a la página principal del administrador y se ocultan los divs que no corresponden a esa página.
- **mostrarPag_principalUsuario:** Se cambia el display de los divs que corresponde a la página principal del usuario y se ocultan los divs que no corresponden a esa página.
- **mostrarLogin():** Se muestra el div que contiene login para iniciar sesión.
- **mostrarLibros():** Se muestran las grafics de las matrices que contiene las dos categorías de libros.
- **mostrrarAutor():** Se muestran los autores cuando se hayan cargado los archivos.
- **buscarAutor():** Se llama al método pre_buscar el cual busca los autores en pre orden y así retorna la información de dicho autor.
- **mostrarPila():** Se muestra la pila según el nombre del libro que ingrese el usuario en el input text.
- **agregarLibro():** Se muestra el div que contiene las etiquetas con las funcionalidades par que el usuario pueda agregar un libro a su biblioteca.
- **mostarBotenesDyA():** Se muestra el div que contiene los botones para poder llamar los métodos que contienen los ordenamientos de forma ascendente y descendente.
- **mostarLibrosA():** Llama al método mostraLibros de la listas, este se encarga de mostrar los libros de forma ascendente, usando el método de ordenamiento de burbuja.
- **mostarLibrosD():** Llama al método mostraLibrosD de la listas, este se encarga de mostrar los libros de forma descendente usando el método de ordenamiento Quicksort..
- **mostrarUusario():** Muestra el div en la cual se va a mostrar la gráfica de los usuarios.
- **mostrarCola():** Muestra el div en el cual se va a pinta la grafica de la cola de espera.
- **mostarTop5():** Muestra el div en el cual se va a mostrar la grafica de la lista doblemente enlazado con los usuarios que tiene la mayor cantidad de libros.

```
class app {

    admin() {
        let nombre = document.getElementById("user").value;
        let password = document.getElementById("password").value;

        if (nombre == "Wilfred" && password == "123") {
            alert("Bienvenido admin")

            let login = document.getElementById("login");
            let opcionesG = document.getElementById("opciones1");
            let opcionesA = document.getElementById("opciones");
            let iniciar = document.getElementById("iniciar");
            let cerrar = document.getElementById("cerrar");

            iniciar.style.display = "none";
            cerrar.style.display = "inline";
            login.style.display = "none";

            opcionesG.style.display = "none";
            opcionesA.style.display = "block";

            //ca.innerHTML='<button type="button" onclick="CargarArchivo.lC.usuario()" >Aceptar</butto

        } else if (CargarArchivo.lC != null) {
            if (CargarArchivo.lC.usuario() == "admin") {
```

Archivo ArbolBinario.js

Clases

- **Autor**
- **NodoArbolBinario**
- **ArolBinario**

Autor

En esta clase se colocan los atributos de un autor, como el dpi, nombre, correo , teléfono, dirección y biografía.

```
class Autor{
    constructor(dpi,nombre,correo,telefono,direccion,biografia){
        this.dpi=dpi
        this.nombre=nombre
        this.correo=correo
        this.telefono=telefono
        this.direccion=direccion
        this.biografia=biografia
    }
}
```

NodoArbolBinario

Esta clase tiene como atributos un autor de tipo Autor, left y right de tipo NodoArbolBinario y un id el cual va a ir aumentando según ingresen los datos.

Esta clase con tiene el método gráfica y crearGrafica, los cuales se encargan de generar la gráfica del árbol binario, el cual cada contiene el nombre del autor, cabe mencionar que el árbol se ordena por medio del nombre del autor.

```
class NodoArbolBinario{
    static cor=1;
    constructor(autor){
        this.autor=autor;
        this.left=null;
        this.right=null;
        this.id=NodoArbolBinario.cor++;


    }
```

ArbolBinario

Esta clase tiene como atributo una raíz de tipo NodoArbolBinario.

La clase tiene el método agregar el cual iguala a la raíz con la función recursiva agregar2. En la función agregar2 se valida que el usuario no este repetido y se ordenan los datos según las reglas del árbol binario de búsqueda.

```
agregar(autor){

    this.raiz=this.agregar2(autor,this.raiz);
}

agregar2(autor,raiz){
    if(raiz==null){
        let nodoArbol= new NodoArbolBinario(autor);
        return nodoArbol;
    }else{
        if (autor.nombre<raiz.autor.nombre) {
            raiz.left=this.agregar2(autor,raiz.left);
        }else if(autor.nombre>raiz.autor.nombre){
            raiz.right=this.agregar2(autor,raiz.right)
        }else{
            console.log("repetido");
        }
    }
    return raiz;
}
```

pre_order

El método pre_order llama al método recursivo pre_order2, para recorrer el árbol en pre_order y así mostrar en el html en forma de tarjetas el nombre del autor.

```
pre_order(raiz){

    this.pre_order2(raiz);

}

pre_order2(raiz){
    if (raiz!=null) {
        console.log("autores "+raiz.autor.nombre)
        //_____________
        let tarjeta = "<div class=\"card\" style=\"width: 9rem; display: inline-block; margin: auto 10px;\" >"
        tarjeta += "<img src=\"https://emser.es/wp-content/uploads/2016/08/usuario-sin-foto.png\" class=\"card-im
        tarjeta += "<div class=\"card-body\">"
        tarjeta += "<p class=\"card-text\">"+raiz.autor.nombre+"</p>"
        tarjeta += "</div>"
        tarjeta += "</div>"

        CargarArchivo.autores.innerHTML+=tarjeta;


        //_____________________________
        this.pre_order2(raiz.left)
        this.pre_order2(raiz.right)
    }
}

/_____________________________
```

pre_bucar2

EL método pre_bucar2 es un método recursivo el cual muestra en el html la información del autor, la búsqueda se realiza por medio del nombre del autor.

```
pre_buscar(raiz,nombre){
    console.log("nnnnnn"+nombre)
    this.pre_buscar2(raiz,nombre);

}

pre_buscar2(raiz,nombre){
    if (raiz!=null) {

        let nombreAutor=document.getElementById("nombreAutor")
        let dpiAutor=document.getElementById("dpiAutor")
        let correoAutor=document.getElementById("correoAutor")
        let telAutor=document.getElementById("telefonoAutor")
        let direccionA=document.getElementById("direccionAutor")
        let bio=document.getElementById("bioAutor")
        //console.log(nombre+"-------"+raiz.nombre)
        if(nombre==raiz.autor.nombre){

            nombreAutor.innerHTML=raiz.autor.nombre;
            dpiAutor.innerHTML=raiz.autor.dpi;
            correoAutor.innerHTML=raiz.autor.correo;
            telAutor.innerHTML=raiz.autor.telefono;
            direccionA.innerHTML=raiz.autor.direccion;
            bio.innerHTML=raiz.autor.biografia;

        }
        //_____________________________
        this.pre_buscar2(raiz.left,nombre)
        this.pre_buscar2(raiz.right,nombre)
    }
```

Graficar

Este método llama al método crearGrafica.

Archivo CargarArchivo

Clase CargarArchivo

Esta clase contiene los métodos para extraer los datos de los archivos de entrada con extensión .json.. Los datos que se obtienen se guardan en la estructura que corresponda para posteriormente poder utilizarlos.

Listado de métodos:

- CargarArchivo

- CargarAutores
- CargarLibros

```
class CargarArchivo {
    static lC;
    static arbol_bi;
    static matrizDis;
    static matrizOrt;
    static autores;
    static listaAs;
    static librosM;
    static colaU;
    static ldoble;
    //_-
    static mTop5;
    static abrirArhivo(event) {
        let auxiliar = "";
        let archivo = event.target.files[0];
        //var lC = new ListaCircular();
        if (archivo) {
            let reader = new FileReader();
            reader.onload = function (e) {
                let contenido = e.target.result;
                auxiliar = contenido;
                //inicializacion
                CargarArchivo.lC = new ListaCircular();
                //cola
                CargarArchivo.colaU=new ColaUsuario()
                CargarArchivo.ldoble=new ListaDoble();
                CargarArchivo.mTop5=document.getElementById("cardsTop5");
                //console.log(contenido);
```

Archivo ColaEspera.js

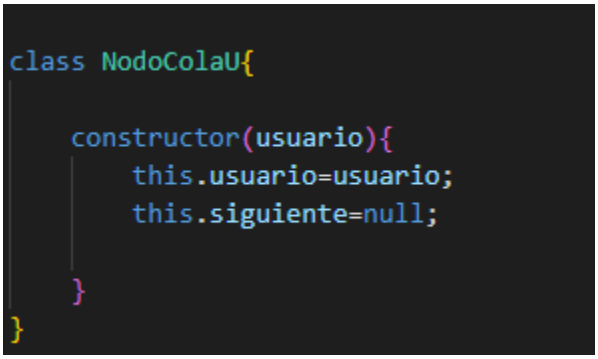
Clase UsuarioE

Esta clase tiene como atributos el usuario, nombre del libro y la cantidad de libros. La cantidad de libros no se recibe como parámetro pero se va incrementando cada vez que el usuario agrega un libro que no se encuentra en la lista de libros y en las matrices.

```
class UsuarioE{

    constructor(usuario,nombre_libro){
        this.usuario=usuario
        this.nombre_libro=nombre_libro
        this.cantidad=1
    }
}
```

NodoColaU

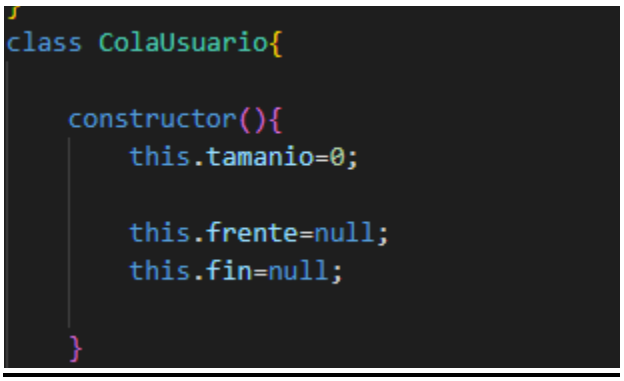
Esta clase tine como atributos un usuario de tipo UsuarioE y un atributo siguiente de tipo NodoColaU.



```
class NodoColaU{

    constructor(usuario){
        this.usuario=usuario;
        this.siguiente=null;

    }
}
```

ColaUsuario

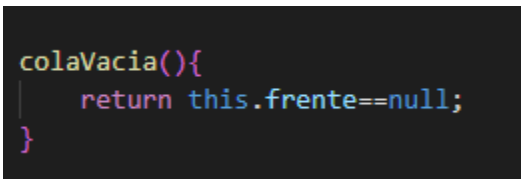
Esta clase como atributos frente e inicio de tipo NodoColaU y un atributo tamanio para ir guardando el tamaño de la cola.



```
class ColaUsuario{

    constructor(){
        this.tamanio=0;

        this.frente=null;
        this.fin=null;

    }
```

Método colaVacia:

Este método retorna true si la cola esta vacía y false si la cola ya contiene un dato.



```
colaVacia(){
    return this.frente==null;
}
```

Método encolar

En este método se van guardando los datos de la forma primero en entrar primero en salir.

```
encolar(usuario){

    let nuevo=new NodoColaU(usuario);
    this.existente(usuario.usuario)
    this.tamanio++;
    if (this.colaVacia()) {
        this.frente=nuevo;
    }else{
        nuevo.siguiente=null;
        this.fin.siguiente=nuevo
    }
    this.fin=nuevo
}
```

Método desencolar

Saca el primer dato que entro a la cola y lo retorna para su uso posterior.

```
desencolar(){
    let auxiliar=null;

    if (!this.colaVacia()) {
        auxiliar=this.frente;

        this.frente=this.frente.siguien
    }

    return auxiliar;
}
```

Método existente

Verifica si el usuario ya esta guardado en la pila, esto para poder aumentar el contador. Si el usuario no se encuentra retorna false.

```
existente(usuario){

    let aux=this.frente;
    while (aux!=null) {
        if (usuario==aux.usuario.usuario) {
            console.log("libro existente")
            aux.usuario.cantidad+=1;

            return true
        }
        aux=aux.siguiente;
    }
    return false;

}
```

Método graficar

Sirve para graficar en la cola y mostrar en el html.

Archivo ListaAscendente.js

NodoAscendente

Esta clase tiene como atributo un libro de tipo Libro, un siguiente de tipo NodoAscendente, tamanio la cual se va incrementando para almacenar el tamaño de la lista.

```
class NodoAscendente {

    constructor(libro) {
        this.libro = libro
        this.siguiente = null;
        this.tamanio = 0;
        this.posicion=0;
    }
}
```

Clase ListaAscendente

Esta clase tiene como atributo inicio a la cual se le asigna el valor null.

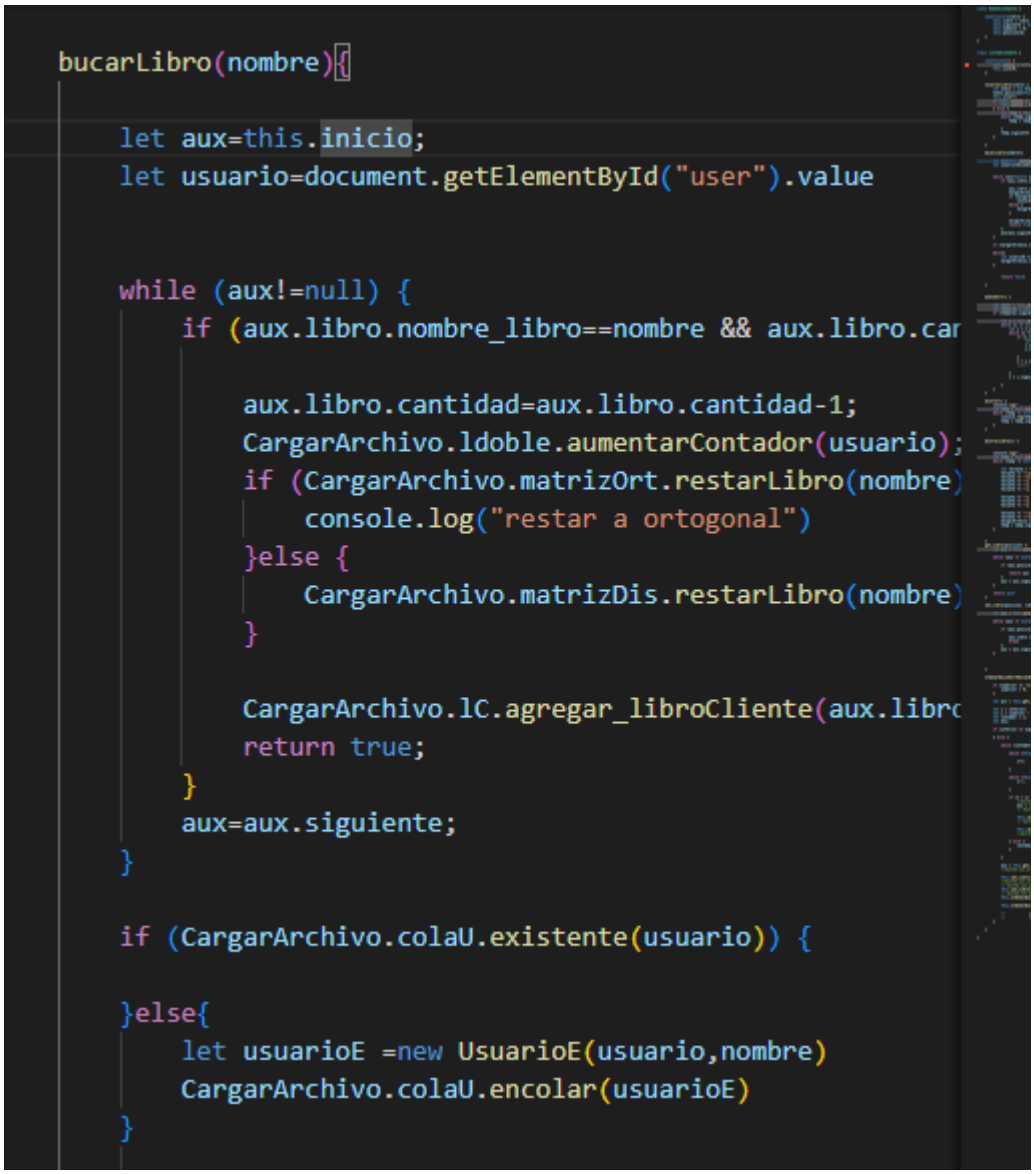
```
class ListaAscendete {

    constructor() {
        this.inicio = null;
        this.size=0;
    }
```

Método insertarLibro

```
insertarLibro(libro) {
    let nuevo = new NodoAscendente(libro)
    nuevo.posicion=this.size;
    this.size++;
    if (this.inicio == null) {
        this.inicio = nuevo;
    } else {
        let temp = this.inicio;
        while (temp.siguiente != null) {
            temp = temp.siguiente;

        }
        temp.siguiente = nuevo;
    }

}
```

Método buscarLibro

Este método compara el nombre que se recibí como parámetro y lo compara como para poder restar el valor a la cantidad.



```
bucarLibro(nombre){

    let aux=this.inicio;
    let usuario=document.getElementById("user").value


    while (aux!=null) {
        if (aux.libro.nombre_libro==nombre && aux.libro.car

            aux.libro.cantidad=aux.libro.cantidad-1;
            CargarArchivo.ldoble.aumentarContador(usuario);
            if (CargarArchivo.matrizOrt.restarLibro(nombre)
                console.log("restar a ortogonal")
            }else {
                CargarArchivo.matrizDis.restarLibro(nombre)
            }

            CargarArchivo.lC.agregar_libroCliente(aux.libro
            return true;
        }
        aux=aux.siguiente;
    }

    if (CargarArchivo.colaU.existente(usuario)) {

    }else{
        let usuarioE =new UsuarioE(usuario,nombre)
        CargarArchivo.colaU.encolar(usuarioE)
    }
```

bubleSort:

En este método se va cambiado de posición por el nombre del libro, esto se hace forma alfabética. Se guarda el numero siguiente y el número actual en dos variables i y j. Se coloca una variable temporal la cual almacena el nombre actual, luego en la variable i se guarda el valor en j y posterior a ello en j se asigna el temporal.

```
bubleSort() {

    let valor = this.inicio;
    let temporal = this.inicio
    if (temporal.siguiente != null && valor != null) {

        let i = this.inicio; //i
        while (i != null) {
            let j = i.siguiente //i+1
            while (j != null) {
                if (i.libro.nombre_libro > j.libro.nombre_libro) {
                    let temporal2 = i.libro.nombre_libro
                    i.libro.nombre_libro = j.libro.nombre_libro
                    j.libro.nombre_libro = temporal2

                }
                j = j.siguiente
                //j++;

            }
            i = i.siguiente//i++

        }
    }
}
```

ordenarQuickSortDescendente2

Para el método Quicksort se utilizan dos métodos, el método getLibro y el método setLibro. El método get_libro retorna el objeto libro que se encuentra en esa posición. El método setLibro recibe la posición y el dato que se quiere colocar en esa posición.

```
ordenarQuickSortDescendente2(inferior, superior) {

    if (superior == -1) {
        superior = 0;
    }

    let piv = this.get_libro(superior).libro.nombre_libro; //quickB[superior];

    let i = inferior;
    let j = superior - 1;
    let contador = 1;
    let aux;

    if (inferior >= superior) {

    } else {

        while (contador == 1) {

            while (this.get_libro(i).libro.nombre_libro > piv) {

                i++;

            }

            while (this.get_libro(j).libro.nombre_libro < piv && j > 0) {
                j--;

            }
```

```
    while (this.get_libro(j).libro.nombre_libro < piv && j > 0) {
        j--;

    }

    if (i < j) {
        //System.out.println("no entra aqui");
        aux = this.get_libro(i).libro;// quickB[i];
        // System.out.println("posicon actual" + aux);

        this.set_libro(i, this.get_libro(j).libro); //quickB[i] = //quickB[j];
        // System.out.println("libron en j" + this.get_libro(j));

        this.set_libro(j, aux);
        //quickB[j] = aux;

    } else {
        contador = 0;
    }

}

aux = this.get_libro(i).libro; //quickB[i];
//System.out.println("aux " + aux);

this.set_libro(i, this.get_libro(superior).libro);// quickB[i] = quickB[superior];
//System.out.println(" superior" + this.get_libro(superior).libro);
// System.out.println("posicion--" + superior);
this.set_libro(superior, aux);//<quickB[superior] = aux;
```

Método graficar

```
mostrarLibros() {

    console.log("_____Libro_____")
    let temp = this.inicio
    while (temp != null) {

        let tarjeta = "<div class=\"card\" style=\"width: 18rem;  margin: auto 10px;\" >"
        tarjeta += "<img src=\"https://img.freepik.com/vector-gratis/libro-abierto-concepto-educacion-lectura_108033-418.jpg\" class=\"card
        tarjeta += "<div class=\"card-body\">"
        tarjeta += "<p class=\"card-text\"> Nombre:" + temp.libro.nombre_libro + "</p>"
        tarjeta += "<p class=\"card-text\">Autor: " + temp.libro.nombre_autor + "</p>"
        tarjeta += "<p class=\"card-text\">ISBN: " + temp.libro.isb + "</p>"

        tarjeta += "<p class=\"card-text\">Paginas: " + temp.libro.paginas + "</p>"
        tarjeta += "<p class=\"card-text\">Categoria: " + temp.libro.categoria + "</p>"
        tarjeta += "<p class=\"card-text\">Cantidad: " + temp.libro.cantidad + "</p>"

        tarjeta += "</div>"
        tarjeta += "</div>"
        CargarArchivo.librosM.innerHTML += tarjeta;
        temp = temp.siguiente
    }
```

Archivo ListaCircular.js

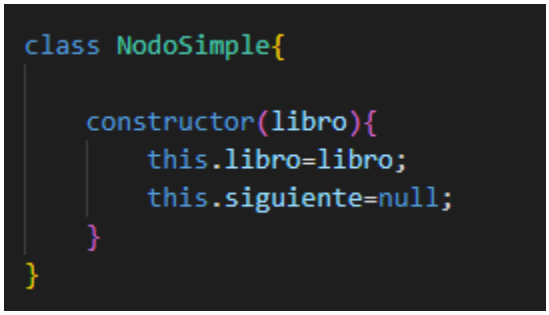
Clase ListaCircular

Esta clase contiene el atributo tamanio y el atributo primero.

```
class ListaSimple{

    constructor(){
        this.tamanio=0;
        this.primero=null;

    }
```

Clase NodoSimple

Esta clase tiene como atributos un libro y un siguiente de tipo NodoSimple.



```
class NodoSimple{

    constructor(libro){
        this.libro=libro;
        this.siguiente=null;
    }
}
```

Método graficar

```
grafica1(nombre) {
    let grafo = "digraph G{\ngraph[size=\"11.70,6.25\"] label=\"Inicio a fin \";\nnode[shape=box];\n";
    let conexion = "";
    let nodos = "";
    let cont = 0;
    let cont2 = 0;
    let contl=0;
    let contl2=0;
    let contAnterior = this.tamanio;
    let temp = this.cabeza;
    let principal = "";

    while(cont<this.tamanio){
        console.log("_____")


        principal += "N" + cont + ";";
        nodos += "N" + cont + "[label=\""+temp.Usuario.dpi +"\\n"+ temp.Usuario.nombre +"\\n"+ temp.Usuario.usua

        if (temp.siguiente != this.cabeza) {
            cont2 = cont + 1;
            console.log("cont2 "+cont2)
            conexion += "N" + cont + "->N" + cont2 + ";\n";
           // conexion += "N" + cont2 + "->N" + cont + ";\n";

        }
        let tempL=temp.lista.primero;

        while (tempL!=null) {

            if (tempL==temp.lista.primero) {
```

Conclusión

El proyecto se realizo con las estructuras de datos lineales y no lineales. Se explico la funcionalidad y la lógica del programa. Al guardar los datos de forma dinámica con las estructuras se tiene un espacio que se adapta a la cantidad de datos que se vayan almacenando. Caso contrario con los arreglos estáticos.