UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERIA

ESTRUCTURA DE DATOS

Nombre: Jonatan Leonel Garcia Arana

Carné: 202000424

MANUAL TECNICO

INTRODUCCION

El presente documento mostrara el manejo de una página web que permite la manipulación de datos tipo música, para presentar de forma sencilla al usuario sus playlist, podcast, artistas y canciones favoritas, el programa esta desarrollado en javascript, html y css.

INFORMACION DESTACADA

Este programa es único dado a base de los requerimientos y necesidades del proyecto que lo requiere

INFORMACION DEL SISTEMA

Requisitos del sistema:

• Windows 10,8,7 (x86 y x64)

• Procesador a 1.6 GHz o superior

• 1 GB (32 bits) o 2 GB (64 bits) de RAM (agregue 512 MB al host si se ejecuta en una máquina virtual)

• 3 GB de espacio disponible en el disco duro

• Disco duro de 5400 RPM

• Tarjeta de vídeo compatible con DirectX 9 con resolución de pantalla de 1024 x 768 o más

Descargar Visual studio Code

link de descarga: https://code.visualstudio.com/download

ver manual de instalación en la página Oficial

INICIANDO EJECUCION DEL PROGRAMA

AGREGAR USUARIOS; método que agrega usuarios o administradores a la lista simple

```
add(dpi1, name1, username1, password1, phone1, admin1) {
  const Nodo = new Cliente(
    dpi1,
    name1,
    username1,
    password1,
    phone1,
    admin1,
    null
  );
  if (!this.head) {
    this.head = Nodo;
  } else {
    let registro = this.head;
    while (registro.next) {
      registro = registro.next;
    }
    registro.next = Nodo;
  }
  this.size++;
  swal(
    "GUARDADO",
    "USUARIO CREADO CORRECTAMENTE" + " " + "Usuario Numero" + this.size,

    "success"
  );
}
```

INGRESO A USUARIOS,este método recibe como parámetros dos valores los cuales los compara con la lista enlazada si encuentra los datos deja ingresar al usuario

```
login_admin(user, pasw) {
    if (!this.size) {
      return "Hola";
    } else {
      let recorrido = this.head;

      while (recorrido) {
        if (recorrido.admin) {
          if (user == recorrido.username && pasw == recorrido.password) {
            document.getElementById("LOGIN-1").style.display = "none";
            document.getElementById("PANTALLA-ADMINISTRADOR").style.display
=
              "block";
              let welcome = 'Bienvenido ' + recorrido.username

              document.getElementById('welcomeb').innerHTML = welcome;
            return "dato encontrado";

          } else {
            recorrido = recorrido.next;
          }
          if (!recorrido) {
            swal("Error", "ADMINISTRADOR NO ENCONTRADO", "error");
            return "error";
          }
        } else {
          if (!recorrido) {
            swal("Error", "ADMINISTRADOR NO ENCONTRADO", "error");
            return "error";
          }
          recorrido = recorrido.next;
        }
      }
    }
  }
```

GRAFICAR LISTA ENLAZADA, este método crea un archivo el cual lo mandamos a graphviz para que cree el grafo

```
graph(idDiv) {
    // creamos la variable del diagraph
    let graphviz =
      'digraph SimpleList{\nnode[shape= box, fillcolor="#FFFFFF", style=
filled];\nbgcolor = "#CD1CED ";\nranksep = 0.5;\nnodesep = 0.5;\nsubgraph
cluster_A{\nlabel = "Clientes";\nbgcolor = "#BC70FC";\nfontcolor
="#3A0964";\nfontsize = 30;\n\n ';


    let current = this.head;
    let i = 1;

    while (current != null) {
      // recorremos la lista hasta que sea null y agregamos un indicie
cliente1
      graphviz += "cliente" + i + '[label="' + current.name + '"];\n';

      i++;
      current = current.next;
    }

    graphviz += "\n";

    //we point the customer nodes

    current = this.head;
    i = 1;

    while (current != null) {
      // aqui de igual manera hasta que sea null y agragemos el indice
      if (current.next != null) {
        graphviz += "cliente" + i + " -> cliente" + (i + 1) + "\n";
      }

      i++;
      current = current.next;
    }

    graphviz += "\n";

    //we aling the nodes

    current = this.head.next;
```

```
    i = 1;

    graphviz += "{rank = same; cliente" + i;

    i++;

    while (current != null) {
      // en esta parte agreagamos el valor de cliente con la posicion i
      graphviz += "; cliente" + i;

      i++;
      current = current.next;
    }

    graphviz += "};\n\n}\n}";

    console.log(graphviz);

    let id = "#" + idDiv;

    d3.select(id) //creamos con d3 el rendeDot y el paragrah
      .graphviz()

      .width(2000)
      .height(1500)
      .zoom(true)
      .fit(true)
      .renderDot(graphviz);
  }
}
```

AÑADIR AL ARBOL BINARIO, este método recibe como parámetro una data la cual hace una restricción con el atributo nombre el cual es el que va ordenando de izquierda o derecha dependiendo que dato es mayor o menor

```
_add(node, data){

      if(data.name < node.getData().name){

          if(node.getLeft() == null){
              let newNode =  new BSTreeNode(data);
              node.setLeft(newNode);

          }else{
              this._add(node.getLeft(), data);
          }


      }else if(data.name > node.getData().name){

          if(node.getRight() == null){
              let newNode =  new BSTreeNode(data);
              node.setRight(newNode);

          }else{
              this._add(node.getRight(), data);

          }

      }else{

          console.log('Se enconstro un elemento igual al que se quiere
insetar, insercción fallida.')
      }
  }
```

AÑADIR DATOS A MATRIZ DISPERSA, Este método agrega datos a la matriz dispersa, recibiendo el mes, dia, artista y canción, la cual evalua el dato de dia en cada columna que se haya creado anteriormente

```
insertar(x, y, obj,cancion) {
      let cell = new NodoMatriz(x, y, obj,cancion);

      let columna = this.colsList.getHeader(y);
      if (columna == null) {
          columna = new NodoHeader(y);
          this.colsList.setHeader(columna);
          columna.access = cell;
      } else if (x < columna.access.x) {
          cell.down = columna.access;
          columna.access.up = cell;
          columna.access = cell;
      } else {
          let aux = columna.access;
          while (aux.down != null) {
              if (x < aux.down.x) {
                  cell.down = aux.down;
                  aux.down.up = cell;
                  aux.down = cell;
                  cell.up = aux;
                  break;
              }
              aux = aux.down;
          }

          if (aux.down == null) {
              aux.down = cell;
              cell.up = aux;
          }
      }

      let row = this.rowsList.getHeader(x);
      if (row == null) {

          row = new NodoHeader(x);
          this.rowsList.setHeader(row);
          row.access = cell;

      } else if (y < row.access.y) {
          cell.next = row.access;
          row.access.prev = cell;
```

```
            row.access = cell;
        } else {
            let aux = row.access;
            while (aux.next != null) {
                if (y < aux.next.y) {
                    cell.next = aux.next;
                    aux.next.prev = cell;
                    aux.next = cell;
                    cell.prev = aux;
                    break;
                }
                aux = aux.next;
            }

            if (aux.next == null) {
                aux.next = cell;
                cell.prev = aux;
            }
        }

        swal("Exito","DATOS CARGADOS","success")
    }
```

AÑADIR BLOQUEADO, Este método es el que añade datos a la cola la cual tiene la funcionalidad de fifo, que significa PRIMERO QUE ENTRA PRIMERO QUE SALE

```
addbloqueados1(name,username,dpi,phone){
   let current = this.head;
   while(current != null){
     if(current.usuario == USUARIO){


       if(this.verificarbloqueado(username)){
         swal("error","USUARIO YA BLOQUEADO", "error")
       }else{




         var newNode = new bloqueados1(name,username,dpi,phone);
         let current3 = current.down;
         if(this.size2 > 0){
           while(current3.next){
```

```
              current3= current3.next;
            }

            current3.next = newNode;
          }else{
            current.down = newNode;
          }
          this.size2++;
          swal("succes","USUARIO BLOQUEADO CON EXITO", "sucess")
          }


      }
      current = current.next;

    }
  }
```

AÑADIR AMIGOS, este método tiene la funcionalidad de añadir amigos la cual se maneja de la manera lifo, que significa PRIMERO EN ENTRAR ULTIMO EN SALIR ya que es una cola

```
addamigos(usuario5,username5,dpi5,numero5){
    let temporalcabeza = this.head;
    while(temporalcabeza != null){
      if(temporalcabeza.usuario ==USUARIO){
        if(this.verificaramigos(username5)){
          swal("oops","El ya es tu amigo", "error")
        }else{
          if(colabloqueados.verificarbloqueado(username5)){
            swal("oops","NO PUEDES AGREGAR A ESTE USUARIO, porque esta
bloqueado, desbloquealo", "error")
          }else{

        var nuevacancion = new amigos1(usuario5,username5,numero5,dpi5)
        var iniciousuarios = temporalcabeza.down;

        if(temporalcabeza.down){
          temporalcabeza.down = nuevacancion;
          temporalcabeza.down.next = iniciousuarios;
        }else{
          temporalcabeza.down = nuevacancion;
        }
        this.size2++;
        swal(
          "GUARDADO",
```

```
          "AMIGO AGREGADO CORRECTAMENTE" + " " + "Amigo numero" +
this.size2,

          "success"
        );



          }



        }
      }

      temporalcabeza = temporalcabeza.next;
    }
```

AÑADIR CANCIONES A UNA LISTA DOBLEMENTE ENLAZADA CIRCULAR, este método cuenta con el código de una lista doblemente enlazada circular, la cual indica que el nodo siguiente de la cola es su cabeza y el nodo anterior de su cabeza es su cola

```
 addcanciones(artista1,cancion2){
    if (this.verificarcancion(cancion2)){
      swal(
        "GUARDADO",
       "LA CANCION YA EXISTE EN LA PLAYLIST",

        "error"
      );
      }else{

    let current = this.head;
    while(current != null){

     if(current.usuario== USUARIO){


       var newNode= new cancion1(artista1,cancion2);
       if(current.list2.head){
        newNode.next =  current.list2.head
        current.list2.head.prev = newNode;
```

```
        current.list2.head = newNode;
       }else{
        current.list2.head = newNode;
        current.list2.tail = newNode;
       }
       current.list2.head.prev= current.list2.tail;
       current.list2.tail.next = current.list2.head;




       this.size2++;
       swal(
         "GUARDADO",
        "cancionguardada" +this.size2,
         "success"
       );
   }
   current = current.next;


 }
}
}
```

FUNCION QUE INGRESA AL LOGIN, Este método cuenta con las restricciones para la validación de usuarios

```
function login() {
  let user = document.getElementById("name1").value;
  let password = document.getElementById("password1").value;
  let check = document.getElementById("check1").checked;

  if (user == "" || password == "") {
    swal("Oops!", "LLENE TODOS LOS CAMPOS", "error");
  } else {
    if (check) {
      if (user == admin.nombre_usuario && password == admin.contrasenia) {
        document.getElementById("LOGIN-1").style.display = "none";
        document.getElementById("PANTALLA-ADMINISTRADOR").style.display =
          "block";
          let welcome = 'Bienvenido ' + admin.nombre_completo

          document.getElementById('welcomeb').innerHTML = welcome;
      } else {
```

```
        digest = sha256(password);
        clientes.login_admin(user, digest);
      }
    } else {
      digest = sha256(password);
      clientes.login_us(user, digest);
    }
  }
}
```

FUNCION CARGAS ARCHIVOS JSON, este método recibe desde el html el archivo json y lo parsea para luego mandarlo a su respectivo método de insertar

```
Function cargar_matriz(e){
  var archivo = e.target.files[0];
  document.getElementById("MusicaFile").files[0]
  if (!archivo) {
    return;
  }
  let lector = new FileReader();
  lector.onload = function (e) {
    let contenido = e.target.result;

    const _clients = JSON.parse(contenido);

    for (const i in _clients) {
      let client1 = _clients[i];
      matrizDispersa.insertar(client1.month,
        client1.day,
        client1.song,
        client1.artist

      );
    }
  };
  lector.readAsText(archivo);

}
document
  .getElementById("MusicaFile")
  .addEventListener("change", cargar_matriz, false);
```