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 y creación de estructura de datos como listas, lista de listas, arboles binarios, arboles avl, tablas hash, arboles merkle y blockchain

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

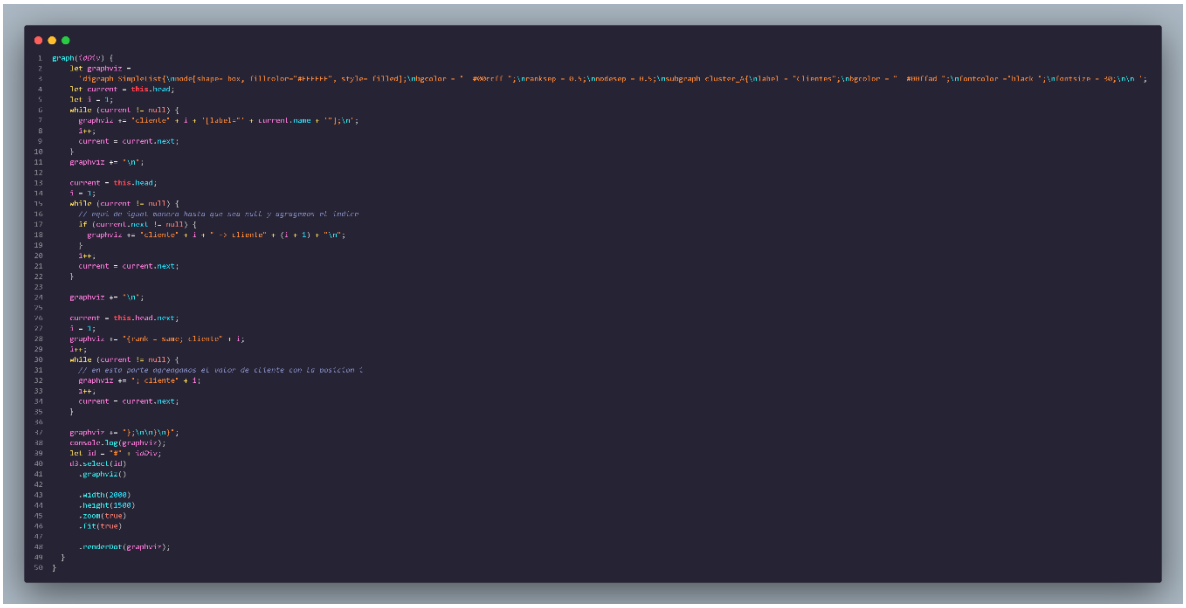
Método add: este método agrega usuarios a una lista enlazada simple

```
add(dpi1, name1, username1, correo1, contraseña1, telefono1) {
    const Nodo = new Cliente(
      dpi1,
      name1,
      username1,
      correo1,
      contraseña1,
      telefono1
    );
    if (!this.head) {
      this.head = Nodo;
    } else {
      let registro = this.head;
      while (registro.next) {
        registro = registro.next;
      }
      registro.next = Nodo;
    }
    this.size++;
    swal(
      "GUARDADO",
      "Cliente Cargado Correctamente" +
        " " +
        "Numero de Clientes: " +
        this.size,
      "success"
    );
  }
```

METODO PARA ENTRAR COMO USUARIO: aquí mostramos la funcionalidad y como recorres la lista para buscar el usuario que quiere ingresar

```
login_us(user, passw) {
   if (!this.size) {
     swal("Error", "NO HAY USUARIOS", "error");
     return true;
   } else {
     let recorrido = this.head;
     while (recorrido) {
       if (user == recorrido.username && passw == recorrido.contraseña) {
         document.getElementById("VENTANA-USUARIO").style.display = "grid";
         document.getElementById("LOGIN").style.display = "none";
         USUARIO33 = recorrido.username;

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

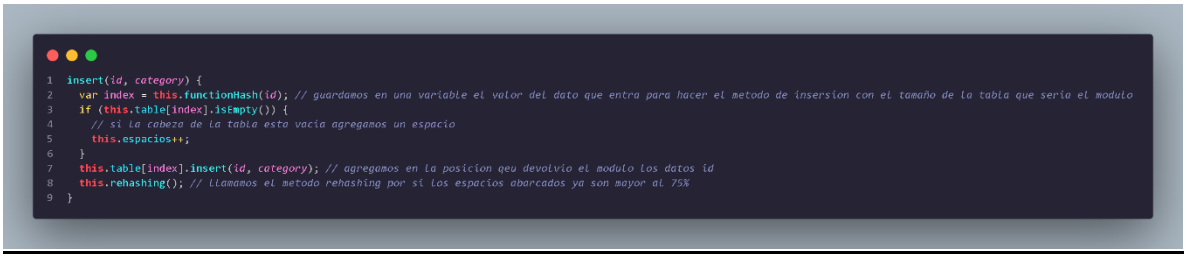
METODO PARA GRAFICAR LISTA ENLAZADA: aquí mostramos como utilizar graphviz



METODO PARA AGREGAR DATOS AL ARBOL BINARIO: aquí mostramos como recorrer el árbol binario utilizando recursividad, ya que dependiendo si es menor o mayor mandamos a llamar otra vez el método, pero con el nuevo dato

```
_add(node, contenido) {
    if (contenido.dni < node.getContenido().dni) {
      // si el dato que entra es menor que el primer dato que esta en cabeza se va a la izquierda

      if (node.getLeft() == null) {
        let newNode = new Arbol(contenido);
        node.setLeft(newNode);
      } else {
        this._add(node.getLeft(), contenido);
      }
    } else if (contenido.dni > node.getContenido().dni) {
      // si el dato es mayor hace lo inverso que el anterior paso

      if (node.getRight() == null) {
        let newNode = new Arbol(contenido);
        node.setRight(newNode);
      } else {
        this._add(node.getRight(), contenido);
      }
    } else {
      console.log("dato igual, no se agrega");
    }
  }
```

METODO PARA AÑADIR A LA TABLA HASH



```
insert(id, category) {
  var index = this.functionHash(id); // guardamos en una variable el valor del dato que entra para hacer el metodo de insersion con el tamaño de la tabla que sería el modulo
  if (this.table[index].isEmpty()) {
    // si la cabeza de la tabla esta vacia agregamos un espacio
    this.espacios++;
  }
  this.table[index].insert(id, category); // agregamos en la posicion qeu devolvio el modulo los datos id
  this.rehashing(); // llamamos el metodo rehashing por si los espacios abarcados ya son mayor al 75%
}
```

METODO PARA HACER EL REHASHING

```
rehashing() {
    var porcentaje = this.espacios / this.size; //aqui obtenemos el dato del porcentaje de datos
    // que seria lso espacios que estamos ocupando en la tabla dividio el tamaño de la tabla
    if (porcentaje > 0.75) {
      var temp = this.table; // guardamos la tabla
      var tempSize = this.size; // y el tamaño de la tabla
      this.size = this.espacios * 5; // por cada ocupacion digamos hay 10 espacios ocupados multiplicamos por 5
      this.table = []; // volvemos la tabla vacia para ordenar la tabla
      for (let i = 0; i < this.size; i++) {
        this.table.push(new Lista()); //volvemos a ordenar la tabla
      }
      this.espacios = 0; // ahora los espacios estan en 0
      for (let i = 0; i < tempSize; i++) {
        // con el tamaño de la tabla anterior
        if (!temp[i].isEmpty()) {
          // si la cabeza es diferente de nula
          var nodo = temp[i].head; // obtenemos la posicion del nodo y el head
          while (nodo != null) {
            // mientras el nodo sea diferente de nullo
            this.insert(nodo.id, nodo.company); // insertamos los nuevos datos
            nodo = nodo.next;
          }
        }
      }
    }
  }
```

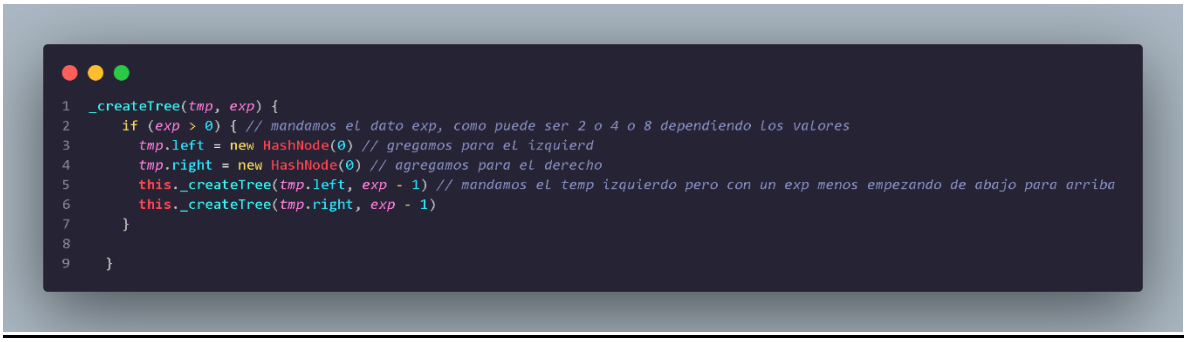
METODO PARA GENERAR BLOQUE

```
generarBloque(pelicula){

    // Debe ir DD-MM-YY-::HH:MM:SS
    var date3 = new Date()//revisar formato
    let date = date3.toLocaleString()



    //Hash Anterior
    var prevHash = "";
    if(this.isEmpty()){
      prevHash = "00" // el primer dato del bloque lleva 00 en el hash
    }else{
      prevHash = this.datablock[this.datablock.length-1].hash; // y si no esta vacio retorna el valor del ultimo dato -1 el hash
    }

    if(this.menor(this.size,this.blockchain.length)){ // si la funcion de menor devuelve true, es porque el arbol merkle le falta el valor del un bloque
      this.blockchain[this.size] = pelicula; // agregamos en el dato  y con eso ya tendriamos el mismo numero de datos
    }else{
      this.blockchain.push(pelicula); // si no solo añadimos
    }
    //generamos el arbol
    this.auth();

    //Data revisar
    var data= this.tophash.hash // aqui obtenemos el dato del tohash

    var nonce = 0;
    var hash = "";

    //prueba de trabajo
    while(!hash.startsWith("00")){
      hash = sha256(this.size+date+prevHash+data+nonce); // aqui encontramos el hash para que empieze en 00, va ir cambiando por el nonce
      nonce += 1;
    }
    var newdata = new Bloque(this.size,date,pelicula,nonce,prevHash,data,hash); // creamos el bloque
    this.datablock.push(newdata) // y lo añadimos
    this.size++;
    dato33 = ""
  }
```

METODO PARA ENCONTRAR EL EXPONENCIAL

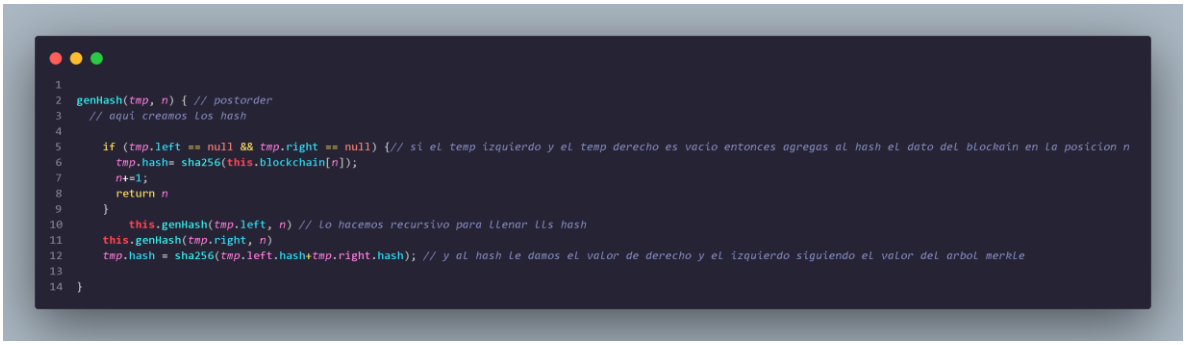
```
auth() {

    var exp = 0
    while (Math.pow(2, exp) < this.blockchain.length) { // mientras el numero exp sea menor que el tamaño de los blockchain este ir aaumentando su valor
      exp += 1 // suma 1 al exponete
    }
    for (var i = this.blockchain.length; i < Math.pow(2, exp); i++) {
      this.blockchain.push(1) // y en este for ingresamos hasta el numero de exp
    }



    this.createTree(exp)

    this.genHash(this.tophash, exp)
  }
```

METODO QUE CREA EL ARBOL MERKLE CON RECURSVIDAD



```
_createTree(tmp, exp) {
    if (exp > 0) { // mandamos el dato exp, como puede ser 2 o 4 o 8 dependiendo los valores
      tmp.left = new HashNode(0) // gregamos para el izquierd
      tmp.right = new HashNode(0) // agregamos para el derecho
      this._createTree(tmp.left, exp - 1) // mandamos el temp izquierdo pero con un exp menos empezando de abajo para arriba
      this._createTree(tmp.right, exp - 1)
    }

  }
```

METODO PARA LOS HASH DEL ARBOL MERKLE

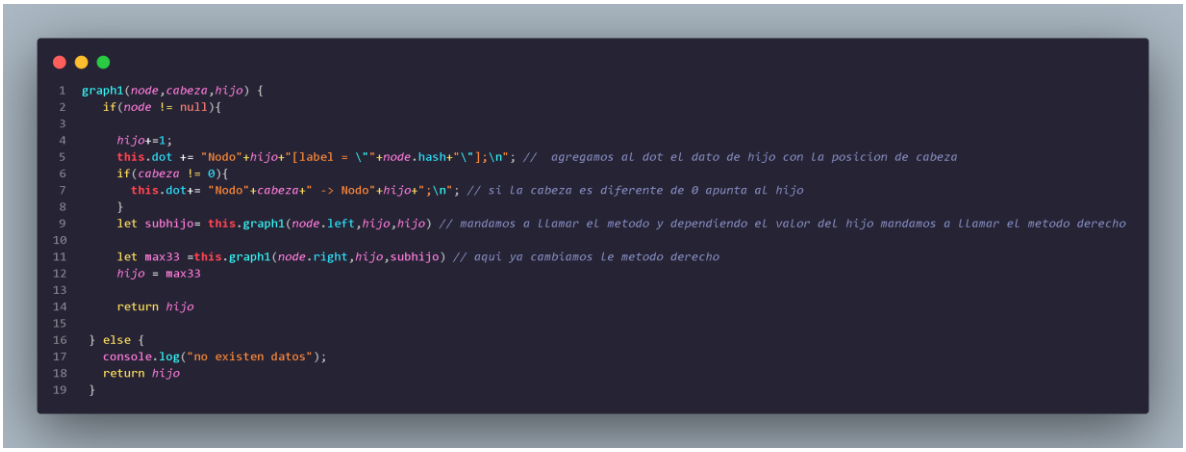


```

genHash(tmp, n) { // postorder
  // aqui creamos los hash

    if (tmp.left == null && tmp.right == null) {// si el temp izquierdo y el temp derecho es vacio entonces agregas al hash el dato del blockain en la posicion n
      tmp.hash= sha256(this.blockchain[n]);
      n+=1;
      return n
    }
       this.genHash(tmp.left, n) // lo hacemos recursivo para llenar lls hash
    this.genHash(tmp.right, n)
    tmp.hash = sha256(tmp.left.hash+tmp.right.hash); // y al hash le damos el valor de derecho y el izquierdo siguiendo el valor del arbol merkle

}
```

METODO PARA ORDEN EXPONENCIAL

```
auth() {

  var exp = 0
  while (Math.pow(2, exp) < this.blockchain.length) { // mientras el numero exp sea menor que el tamaño de los blockchain este ir aaumentando su valor
    exp += 1 // suma 1 al exponete
  }
  for (var i = this.blockchain.length; i < Math.pow(2, exp); i++) {
    this.blockchain.push(1) // y en este for ingresamos hasta el numero de exp
  }
```

METODO PARA GRAFICAR EL ARBOL MERKLE



```
graph1(node,cabeza,hijo) {
  if(node != null){

    hijo+=1;
    this.dot += "Nodo"+hijo+"[label = \""+node.hash+"\"];\n"; //  agregamos al dot el dato de hijo con la posicion de cabeza
    if(cabeza != 0){
      this.dot+= "Nodo"+cabeza+" -> Nodo"+hijo+";\n"; // si la cabeza es diferente de 0 apunta al hijo
    }
    let subhijo= this.graph1(node.left,hijo,hijo) // mandamos a llamar el metodo y dependiendo el valor del hijo mandamos a llamar el metodo derecho

    let max33 =this.graph1(node.right,hijo,subhijo) // aqui ya cambiamos le metodo derecho
    hijo = max33

    return hijo

} else {
  console.log("no existen datos");
  return hijo
}
```