

Escuela de Ciencias y Sistemas

Estructura de Datos

Proyecto – Fase 1
Manual Técnico

César André Ramírez Dávila

202010816

Fecha: 24/08/2022

INDICE

<i>Introducción</i>	3
<i>Objetivos</i>	3
<i>Requerimientos</i>	4
<i>Creación del programa</i>	5
1. Librerías	5
1.1. Archivos de cabecera	5
1.2. Archivo de entrada.....	5
2. <i>Main.cpp</i>	7
2.1. Funciones declaradas	7
2.2. Nodos	7
2.3. Menú principal.....	8
2.4. Carga Masiva.....	8
2.5. Registros JSON	10
3. <i>Registro Usuarios.....</i>	12
4. <i>Login.....</i>	13
4.1. Menu Login	13
4.2. Editar información	14
4.3. Eliminar Cuenta.....	15
4.4. Ver Tutorial	17
4.5. Ver Articulos de Tienda	17
4.6. Realizar Movimientos.....	18
5. <i>Reportes</i>	19
5.1. Estructuras Utilizadas.....	19
5.2. Lista de Usuarios Ordenadas	21
5.3. Lista de Articulos Ordenadas	22

Introducción

El presente documento describe los aspectos técnicos informáticos del programa. El documento familiariza al desarrollador que hace uso del lenguaje C++ con temas como: uso de nodos, listas, ordenamientos, ciclos repetitivos, cargas de archivos con muchos datos, registros, graficas.

Aprendiendo como usar el algoritmo de hash de seguro de 256 bits para la seguridad en las contraseñas de los usuarios.

Objetivos

- Instruir el uso adecuado del Sistema de información, para el acceso adecuado en el uso de este, mostrando los pasos de desarrollo del programa, así como la descripción de las funciones y métodos usados para la realizacion del programa.
- Comprender uso de estructuras de datos en el lenguaje C++
- Obtener mayor conocimiento en el lenguaje C++

El presente manual está enfocado en el lenguaje de programación C++.

Requerimientos

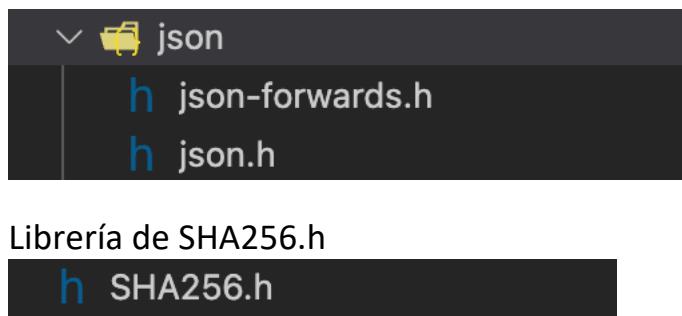
- La aplicación puede ser ejecutada en cualquier sistema operativo que tenga configurado el compilador de C++ en el sistema, en este caso el compilador usado es: mingw-w64.
- IDE recomendado: Visual Estudio Code
- Equipo Intel Pentium o superior
- Espacio en el disco duro, al menos 500 mb
- Memoria ram recomendada 2gb

Creación del programa

1. Librerías

Las siguientes librerías son requeridas para hacer la carga de un archivo. json y encriptamiento con SHA256.

Se necesita la siguiente carpeta de json dentro del proyecto, para poder hacer uso de la librería.



Librería de SHA256.h

h SHA256.h

1.1. Archivos de cabecera

```
1 #include<iostream>
2 #include <curses.h>
3 #include<fstream>
4 #include<string.h>
5 #include<string>
6 #include<stdlib.h>
7 #include <sstream>
8 #include "json/json.h"
9 #include "jsoncpp.cpp"
10 #include "SHA256.h"
```

1.2. Archivo de entrada

Se cuenta con un archivo. Json para la información que se debe almacenar dentro del programa como una carga masiva, la estructura es la siguiente:

```

1 ↴ {
2 ↴   "usuarios": [
3 ↴     { "nick": "andre", "password": "card", "monedas": "200", "edad": "20" },
4 ↴     { "nick": "luis", "password": "elpana", "monedas": "100", "edad": "40" },
5 ↴     { "nick": "angel", "password": "siquea", "monedas": "50", "edad": "21" }
6 ↴   ],

```

Se tiene un arreglo de usuario, este dentro contiene el nick, contraseña, monedas y la edad.

```

20   "articulos": [
21     { "id": "1", "categoria": "legendario", "precio": "500", "nombre": "barco1", "src": "link1" },
22     { "id": "2", "categoria": "epic", "precio": "250", "nombre": "barco2", "src": "link2" },
23     { "id": "3", "categoria": "basico", "precio": "100", "nombre": "barco3", "src": "link3" },
24     { "id": "4", "categoria": "epic", "precio": "250", "nombre": "barco4", "src": "link4" },
25     { "id": "2", "categoria": "epic", "precio": "250", "nombre": "barco5", "src": "link5" },
26     { "id": "6", "categoria": "raro", "precio": "500", "nombre": "barco6", "src": "link6" },
27     { "id": "7", "categoria": "mitico", "precio": "250", "nombre": "barco7", "src": "link7" },
28     { "id": "8", "categoria": "especial", "precio": "100", "nombre": "barco8", "src": "link8" },
29     { "id": "9", "categoria": "raro", "precio": "250", "nombre": "barco9", "src": "link9" },
30     { "id": "10", "categoria": "mitico", "precio": "250", "nombre": "barco10", "src": "link10" }
31   ],

```

Se tiene un arreglo de artículos, en el cual dentro contiene el id, categoría, precio, nombre y un enlace a una imagen.

```

32   "tutorial": {
33     "ancho": "20",
34     "alto": "30",
35     "movimientos": [
36       { "x": "1", "y": "6" },
37       { "x": "3", "y": "9" },
38       { "x": "5", "y": "1" },
39       { "x": "8", "y": "3" },
40       { "x": "1", "y": "0" }
41   ],
42 }

```

Por último tenemos un arreglo dentro de otro, donde primero viene las dimensiones del tablero y seguido los movimientos realizados.

2. Main.cpp

En este caso el proyecto fue trabajado en un unico archivo llamado main, por lo cual todas las funciones son únicamente para este mismo.

2.1. Funciones declaradas

```
15 | void MenuPrincipal();
16 | void cargamasiva();
17 | void registrousuario();
18 | void login();
19 | void sub_login(string nombreuser, string contra,int edad, int monedas);
20 | void reportes();
21 | void sub_reportes();
22 | void registro_usuario(string nombreuser, string contra,int monedas, int edad, string contracifrada);
23 | void registro_usuarioJ(string nombreuser, string contra,int monedas, int edad, string contracifrada);
24 | void registro_articulos(string categoriaarticulo, string nombrearticulo, int precioarticulo, int idarticulo, string srcarticulo);
25 | void registroTutorial(int x, int y);
26 | void insertarPila(int movx, int movy);
27 | void verTutorial();
28 | void lista_usuarios();
29 | void lista_usuariosordenada();
30 | void eliminarCuenta(string nombreuser);
31 | void editar_info(string nombreuser, int edad, string contra);
32 | void modificarNick(string nombreuser);
33 | void modificarEdad(int edad);
34 | void modificarContra(string contra);
35 | void GraficolistaCDobleEnlace();
36 | void GraficoTutorial();
37 | void movimientos(string nombreuser);
38 | void desplegarPila();
39 | void GraficoMovimientos(string nombreuser, string salida);
40 | void Mostrar_Tienda(int monedas);
41 | void ordenarPrecioASC();
42 | void ordenarPrecioDESC();
43 | void GraficolistadeListas();
44 | void vaciarPila();
```

Todas estas funciones son declaradas para posteriormente hacer el llamado, algunas necesitan que se envíen parámetros para poder ser completadas.

2.2. Nodos

```
51     struct nodo{
52         string nombreuser, contra, contracifrada;
53         int monedas,edad;
54         nodo *anterior;
55         nodo *siguiente;
56     } *primero=NULL, *ultimo=NULL;
57
58     struct NodoArticulos{
59         int idArticulo, precioArticulo;
60         string nombreArticulo,SRCArticulo;
61         NodoArticulos* siguienteArtic;
62     }*primeroArticulos, *ultimoArticulos;
63
64     struct NodoCategoria{
65         string categoria;
66         int indice;
67         NodoCategoria* siguienteCA;
68         NodoArticulos* abajo;
69     }*primeroCategoria, *ultimoCategoria;
70
71     struct nodoCola{
72         int x,y;
73         nodoCola* siguienteCola;
74     } *primeroCola, *ultimoCola;
75
76     struct nodoPila{
77         int x,y;
78         nodoPila* siguientePila;
79     } *primeroPila;
```

Los siguientes son nodos, para lista doblemente enlazada, lista de listas, cola y una pila. Cada uno identificado que información guarda.

2.3. Menú principal

```
120 void MenuPrincipal(){
121     int op=0;
122     do {
123         cout<<"***** Menu *****" << endl;
124         cout<<"* 1. Carga Masiva          *" << endl;
125         cout<<"* 2. Registrar Usuario      *" << endl;
126         cout<<"* 3. Login                  *" << endl;
127         cout<<"* 4. Reportes                *" << endl;
128         cout<<"* 5. Salir del juego        *" << endl;
129         cout<<"*****" << endl;
130         cin>>op;
131         switch(op){
132             case 1:
133                 cargamasiva();
134                 break;
135             case 2:
136                 registrousuario();
137                 break;
138             case 3:
139                 login();
140                 break;
141             case 4:
142                 reportes();
143                 break;
144             case 5:
145                 cout << "\nFin del programa\n";
146                 break;
147             default:
148                 cout << "\nIngrese una opcion correcta\n\n";
149                 break;
150         }
151     }while (op!=5);
152 }
```

Se tienen 4 opciones de acciones que se pueden realizar, se cuenta con un ciclo que se repite hasta que la opción sea “Salir del juego”.

2.4. Carga Masiva

```
154 void cargamasiva(){
155     ifstream archivo;
156     string ruta;
157     string texto;
158     string nombreuser,contra,monedas,edad;
159     string idarticulo, categoriaarticulo,precioarticulo,nombrearticulo,srccarticulo;
160     string alt, anch, x1, y1;
161     cout<<"Ingrese la ruta del archivo " << endl;
162     cin.ignore();
163     getline(cin,ruta);
164     archivo.open("informacion.json", ios::in);
165     //archivo.open(ruta.c_str(), ios::in);
166     if(archivo.fail()){
167         cout<<"\nNo se pudo abrir el archivo\n" << endl;
168     }
```

Previamente con la librería para JSON cargada correctamente, procedemos a hacer la lectura del archivo, pidiendo la ruta para el archivo y luego haciendo una condición de error al archivo.

```

170     while (!archivo.eof())
171     {
172         Json::Reader reader;
173         Json::Value obj;
174         reader.parse(archivo, obj);
175         const Json::Value& usuariosJ = obj["usuarios"];
176         for (int i = 0; i < usuariosJ.size(); i++){
177             //cout << "\nNick: " << usuariosJ[i]["nick"].asString();
178             nombreuser = usuariosJ[i]["nick"].asString();
179             //cout << "\nPass: " << usuariosJ[i]["password"].asString();
180             contra = usuariosJ[i]["password"].asString();
181             string encriptado = SHA256::cifrar(contra);
182             //cout<<"El cifrado sha es : "<<encriptado<<endl;
183             //cout << "\nMonedas: " << usuariosJ[i]["monedas"].asString();
184             monedas = usuariosJ[i]["monedas"].asString();
185             //cout << "\nEdad: " << usuariosJ[i]["edad"].asString();
186             edad = usuariosJ[i]["edad"].asString();
187             std ::string edadi = edad;
188             std ::string monedasi = monedas;
189             int eddi = std::stoi(edadi);
190             int monedi = std::stoi(monedasi);
191             registro_usuarioJ(nombreuser,contra,monedi,eddi,encriptado);
192         }
193     }

```

Si no existen errores al abrir el archivo, procedemos a recorrer el archivo JSON, en usuario leyendo el Nick, contraseña, pero esta antes de ser enviada es cifrada con SHA256, monedas y la edad. Luego estas se mandan a una función llamada `registro_UsuarioJ`.

```

194     const Json::Value& articulosJ = obj["articulos"];
195     for (int i = 0; i < articulosJ.size(); i++){
196         //cout << "\nID: " << articulosJ[i]["id"].asString();
197         idarticuloo = articulosJ[i]["id"].asString();
198         //cout << "\nCategoria: " << articulosJ[i]["categoria"].asString();
199         categoriarticulo = articulosJ[i]["categoria"].asString();
200         //cout << "\nPrecio: " << articulosJ[i]["precio"].asString();
201         precioarticoloo = articulosJ[i]["precio"].asString();
202         articulosJ[i]["precio"].asString();
203         //cout << "\nNombre: " << articulosJ[i]["nombre"].asString();
204         nombrearticulo = articulosJ[i]["nombre"].asString();
205         //cout << "\nSRC: " << articulosJ[i]["src"].asString();
206         srcarticulo = articulosJ[i]["src"].asString();
207         std ::string iarticulo = idarticuloo;
208         std ::string precioarticul = precioarticoloo;
209         int precioarticulo = std::stoi(precioarticul);
210         int idarticulo = std::stoi(iarticulo);
211         registro_articulos(categoriarticulo,nombrearticulo,precioarticulo,idarticulo,srcarticulo);
212         //cout << endl;
213     }

```

Repetimos el mismo proceso para recorrer los artículos y estos se mandan a la función llamada `registro_articulos`.

```

215     const Json::Value& tutorialJ = obj["tutorial"];
216     //cout <<"\nAncho: "<<tutorialJ["ancho"].asString();
217     anch = tutorialJ["ancho"].asString();
218     //cout <<"\nAlto: "<<tutorialJ["alto"].asString();
219     alt = tutorialJ["alto"].asString();
220     std ::string ancho = anch;
221     std ::string alto = alt;
222     int x = std::stoi(ancho);
223     int y = std::stoi(alto);
224     registroTutorial(x,y);
225     //cout <<"\nMovimientos: ";
226     const Json::Value& movimientosJ = tutorialJ["movimientos"];
227     for(int i = 0; i < movimientosJ.size(); i++){
228         //cout << "\nX: " << movimientosJ[i]["x"].asString();
229         x1 = movimientosJ[i]["x"].asString();
230         //cout << " Y: " << movimientosJ[i]["y"].asString();
231         y1 = movimientosJ[i]["y"].asString();
232         int x = std::stoi(x1);
233         int y = std::stoi(y1);
234         registroTutorial(x,y);
235     }
236     cout << "\n";
237     cout << "\nArchivo cargado con exito\n" << endl;
238     break;
239 }
240 archivo.close();
241 return;
242 }
```

Por último, el tutorial se recorre de otra forma, debido a que primero trae las dimensiones del tablero y luego el arreglo de los movimientos. De igual forma estos datos son mandados a la función registroTutorial.

2.5. Registros JSON

registro_usuarioJ();

```

414 void registro_usuarioJ(string nombreuser, string contra, int monedas ,int edad, string contracifrada){
415     nodo *actual = new nodo();
416     actual = primero;
417     bool encontrado = false;
418
419     if(primer != NULL){
420         do{
421             if(actual->nombreuser==nombreuser){
422                 encontrado = true;
423             }
424             actual = actual->siguiente;
425         }while(actual!=primer && encontrado != true);
426     }
427     if(primer!= NULL && encontrado==false){
428         if(actual->nombreuser!=nombreuser){
429             nodo *nuevo = new nodo();
430             nuevo->nombreuser = nombreuser;
431             nuevo->contra = contra;
432             nuevo->monedas = monedas;
433             nuevo->edad = edad;
434             nuevo->contracifrada = contracifrada;
435
436             if (primer==NULL) {
437                 primer=nuevo;
438                 ultimo=nuevo;
439                 primero->siguiente=primer;
440                 primero->anterior=ultimo;
441             }else{
442                 ultimo->siguiente=nuevo;
443                 nuevo->anterior=ultimo;
444                 nuevo->siguiente=primer;
445                 ultimo=nuevo;
446                 primero->anterior=ultimo;
447             }
448         }
449     }
```

Antes de registrar al segundo usuario, primero se verifica si el nick anterior no es el mismo, en este caso lo agrega, así sucesivamente con los demás usuarios que vayan asignando, si encuentra un repetido este no se registra.

registro_articulos();

```
1201 void registro_articulos(string categoria, string nombre, int precio, int id, string srcarticulo){  
1202     NodoCategoria* nodoCategoria = new NodoCategoria();  
1203     nodoCategoria->categoria = categoria;  
1204     NodoCategoria* aux = new NodoCategoria();  
1205  
1206     NodoArticulos* nodoArticulos = new NodoArticulos();  
1207     nodoArticulos->nombreArticulo = nombre;  
1208     nodoArticulos->precioArticulo = precio;  
1209     nodoArticulos->idArticulo= id;  
1210     nodoArticulos->SRCArticulo = srcarticulo;  
1211  
1212     aux = primeroCategoria;  
1213     //nodoArticulos = primeroArticulos;  
1214  
1215     bool encontrado = false;  
1216  
1217     if(primerоСategoria == NULL){  
1218         nodoCategoria->indice++;  
1219         primeroCategoria = nodoCategoria;  
1220         ultimoCategoria = nodoCategoria;  
1221  
1222         primeroCategoria->abajo = nodoArticulos;  
1223         primeroArticulos = nodoArticulos;  
1224         ultimoArticulos = nodoArticulos;  
1225     }else{
```

Este registro requiere uso de dos listas, una lista para Artículos y otra lista para las categorías. Debido a que si hay categorías repetidas, el nombre y los demás atributos del artículo deben almacenarse en esa categoría.

```
1225     }else{  
1226         if(!buscarRepetido(nodoCategoria->categoria)){  
1227             NodoCategoria* temp = new NodoCategoria();  
1228             temp = primeroCategoria;  
1229             int aux = 1;  
1230             while (temp!=NULL)  
1231             {  
1232                 aux++;  
1233                 temp = temp->siguienteCA;  
1234             }  
1235             nodoCategoria->indice = aux;  
1236  
1237             ultimoCategoria->siguienteCA = nodoCategoria;  
1238             ultimoCategoria = nodoCategoria;  
1239  
1240             ultimoCategoria->abajo = nodoArticulos;  
1241             primeroArticulos = nodoArticulos;  
1242             ultimoArticulos = nodoArticulos;  
1243         }else{  
1244             ultimoArticulos->siguienteArtic = nodoArticulos;  
1245             ultimoArticulos = nodoArticulos;  
1246         }  
1247     }
```

Con una función que busca si la categoría es existente, para almacenarla en esta.

```
registroTutorial();
```

```
554     void registroTutorial(int x, int y){  
555         nodoCola* nuevoCola = new nodoCola();  
556         nuevoCola->x = x;  
557         nuevoCola->y = y;  
558         if(primerоЮCola==NULL){  
559             primeroCola = nuevoCola;  
560             primeroCola->siguienteCola = NULL;  
561             ultimoCola = primeroCola;  
562         }else{  
563             ultimoCola->siguienteCola = nuevoCola;  
564             nuevoCola->siguienteCola = NULL;  
565             ultimoCola = nuevoCola;  
566         }  
567     }  
568 }
```

Para este registro es una cola, en la primera posición guardamos el alto y ancho del tablero, y en las siguientes posiciones los movimientos, en una forma de (X,Y).

3. Registro Usuarios

```
244     void registrousuario(){  
245  
246         string nombreuser, contra;  
247         int monedas, edad;  
248         cout << "Ingresa el nombre de usuario: "<<endl;  
249         cin >> nombreuser;  
250         cout << "Ingresa la contraseña: "<<endl;  
251         cin >> contra;  
252         //cout << "Ingresa las monedas actual: "<<endl;  
253         //cin >> monedas;  
254         monedas = 0;  
255         cout << "Ingresa la edad: "<<endl;  
256         cin >> edad;  
257         string encryptado = SHA256::cifrar(contra);  
258         //cout<<"El cifrado sha es : "<<encryptado<<endl;  
259         registro_usuario(nombreuser,contra,monedas, edad,encryptado);  
260         cout<<"\n";  
261     }
```

Para el registro de usuarios de forma individual se tiene otra función que pide los datos del nick, contraseña y esta es cifrada antes de ser enviada, por ser registro nuevo se le otorgan 0 monedas y por último la edad.

Esta información es mandada a un registro donde está la lista circular que se conecta con los usuarios cargados con un archivo o viceversa.

4. Login

```
263 void login(){  
264     nodo* actual = new nodo();  
265     actual = primero;  
266     bool encontrado = false;  
267     string nodoBuscado;  
268     string usuariob, contrab;  
269     char caracter;  
270     cout << "Ingrese su usuario: "<<endl;  
271     cin >> usuariob;  
272     cout << "Ingrese su contraseña: "<<endl;  
273     cin >> contrab;  
274     string cifrada = SHA256::cifrar(contrab);  
275     //cout<<"El cifrado sha es : "<<cifrada<<endl;  
276  
277     cout<<"\n";  
278     if(primer!=NULL){  
279         do{  
280             //cout<<"El cifrado sha a comparar es : "<<actual->contracifrada<<endl;  
281             if(actual->nombreuser==usuariob && actual->contracifrada==cifrada){  
282                 encontrado = true;  
283                 cout<<" Datos correctos"<<endl;  
284                 userlogin = actual->nombreuser;  
285                 sub_login(actual->nombreuser, actual->contra,actual-> edad, actual->monedas);  
286             }  
287  
288             actual = actual->siguiente;  
289         }while(actual!=primero && encontrado != true);  
290  
291         if(!encontrado){  
292             cout << "\nUsuario o contraseña incorrectos\n\n";  
293         }  
294     }else{  
295         cout << "\nNo existe el usuario en la lista\n\n";  
296     }  
297 }  
298 }
```

En esta parte se piden que ingresen los cambios de usuario y contraseña, la contraseña colocada se vuelve a cifrar, luego se hace una búsqueda para validar que los datos existen, sino sale un mensaje de error.

4.1. Menu Login

```
695 void sub_login(string nombreuser, string contra,int edad, int monedas){  
696     int op1=0;  
697     char prueba;  
698     do {  
699         cout<<"***** Usuario *****" <<endl;  
700         cout<<"* 1. Editar Informacion      *" <<endl;  
701         cout<<"* 2. Eliminar Cuenta       *" <<endl;  
702         cout<<"* 3. Ver tutorial          *" <<endl;  
703         cout<<"* 4. Ver articulos de tienda *" <<endl;  
704         cout<<"* 5. Realizar movimientos   *" <<endl;  
705         cout<<"* 6. Cerrar sesión          *" <<endl;  
706         cout<<"*****" <<endl;  
707         cin>>op1;  
708         cout<<"\n";
```

Luego de comprobar las credenciales y existan, se manda a un nuevo menú diseñado para el usuario, donde puede editar información, eliminar cuenta, ver el tutorial del juego, ver artículos de la tienda, realizar movimientos en el juego y cerrar sesión.

4.2. Editar información

```
744 void editar_info(string nombreuser, int edad, string contra){  
745     int opcion;  
746     cout<<"\n";  
747     cout<<"Elija lo que quiere editar: "<<endl;  
748     cout<<"1. Editar Nick"<<endl;  
749     cout<<"2. Editar Edad"<<endl;  
750     cout<<"3. Editar Contraseña"<<endl;  
751     cout<<"4. Regresar"<<endl;  
752     cin>>opcion;  
753  
754     switch (opcion){  
755         case 1:  
756             cout<<"Opcion 1";  
757             modificarNick(nombreuser);  
758             break;  
759         case 2:  
760             cout<<"Opcion 2";  
761             modificarEdad(edad);  
762             break;  
763         case 3:  
764             cout<<"Opcion 3";  
765             modificarContra(contra);  
766             break;  
767         case 4:  
768             cout<<"\n";  
769             break;  
770         default:  
771             cout << "\nIngrese una opcion correcta\n\n";  
772             break;  
773     }  
774 }  
775 }
```

El usuario puede elegir si deseas cambiar su nick, edad o contraseña

```
779 void modificarNick(string userb){  
780     nodo* actual = new nodo();  
781     actual = primero;  
782     bool encontrado = false;  
783     if(primer!=NULL){  
784         do{  
785             if(actual->nombreuser==userb){  
786                 cout << "\n Ingrese el nuevo Nick: ";  
787                 cin >> actual->nombreuser;  
788                 cout << "\n Para efectuar los cambios debe cerrar sesión e iniciar con el nuevo usuario\n\n";  
789                 encontrado = true;  
790             }  
791             actual = actual->siguiente;  
792         }while(actual!=primero && encontrado != true);  
793     }  
794 }
```

Se hace una búsqueda de la posición del usuario y se le cambiar el valor actual por el nuevo nick que ingrese.

```

901 void modificarEdad(int edad){
902     nodo* actual = new nodo();
903     actual = primero;
904     bool encontrado = false;
905     if(primer !=NULL){
906         do{
907             if(actual->edad==edad){
908                 cout << "\n Ingrese la nueva Edad: ";
909                 cin >> actual->edad;
910                 cout << "\n Para efectuar los cambios debe volver a iniciar sesión\n\n";
911                 encontrado = true;
912             }
913             actual = actual->siguiente;
914         }while(actual!=primer && encontrado != true);
915     }
916 }
```

Se hace una busqueda de la posicion del usuario y se le cambiar el valor actual por la nueva edad que ingrese.

```

919 void modificarContra(string contra){
920     nodo* actual = new nodo();
921     actual = primero;
922     string cambiocontra;
923     bool encontrado = false;
924     if(primer !=NULL){
925         do{
926             if(actual->contra==contra){
927                 cout << "\n Ingrese la nueva Contraseña: ";
928                 cin>>cambiocontra;
929                 //cin >> actual->contra;
930                 string encriptado = SHA256::cifrar(cambiocontra);
931                 actual->contracifrada = encriptado;
932                 actual->contra = cambiocontra;
933                 cout << "\n Para efectuar los cambios debe cerrar sesión e iniciar con la nueva contraseña\n\n";
934                 encontrado = true;
935             }
936             actual = actual->siguiente;
937         }while(actual!=primer && encontrado != true);
938     }
939 }
```

Se hace una busqueda de la posicion del usuario y se le cambiar el valor actual por la nueva contraseña que ingrese, seguidamente volviendo a encriptar la nueva contraseña.

4.3. Eliminar Cuenta

```

940 void eliminarCuenta(string userbuscado){
941     nodo* actual = new nodo();
942     actual = primero;
943     nodo* anterior = new nodo();
944     anterior = NULL;
945     bool encontrado = false;
946     string opc;
947     cout<<"Desea eliminar su cuenta permanentemente [y/s] : " <<endl;
948     cin>>opc;
```

Al seleccionar esta opcion, se le pregunta al usuario si está seguro de eliminar su cuenta.

```

949     if (opc == "y"){
950         if(primer !=NULL){
951             do{
952                 if(actual->nombreuser==userbuscado){
953                     if(actual==primer){
954                         primer = primer->siguiente;
955                         primer->anterior = ultimo;
956                         ultimo->siguiente = primer;
957                     }else if(actual==ultimo){
958                         ultimo = anterior;
959                         ultimo->siguiente = primer;
960                         primer->anterior = ultimo;
961                     }else{
962                         anterior->siguiente = actual->siguiente;
963                         actual->siguiente->anterior = anterior;
964                     }
965                     cout << "\nLa cuenta ha sido eliminada\n\n";
966                     encontrado = true;
967                 }
968                 anterior = actual;
969                 actual = actual->siguiente;
970             }while(actual!=primer && encontrado != true);
971         }
972         cout<<"Cerrando la sesión"<<endl;
973         cout<<"\n";
974         MenuPrincipal();
975     }

```

Al seleccionar si, se hace la busqueda de la posicion del usuario, luego se elimina de la lista, se cierra sesión automaticamente y vuelve al menu principal.

```

976     if (opc == "s"){
977         cout<<"\nRegresando\n"<<endl;
978         return;
979     }
980     else{
981         cout<<"Ingrese una opcion valida"<<endl;
982     }
983 }

```

Si elige la segunda opcion unicamente regresa el menu.

4.4. Ver Tutorial

```
● 579 ˜ void verTutorial(){
580      nodoCola* actualCola = new nodoCola();
581      cout<<"    Tablero\n";
582      actualCola = primeroCola->siguienteCola;
583 ˜ if(primerCola!=NULL){
584      cout<<"\tAncho: " << primeroCola->x<<"\n";
585      cout<<"\tAlto: "<<primeroCola->y<<"\n";
586      cout<<"    Movimientos: "<<endl;
587      cout<<"\t";
588 ˜˜ while(actualCola!=NULL){
589 ˜˜˜ if(actualCola!= ultimoCola){
590      cout<< "(" << actualCola->x<<","<<actualCola->y<< ")--";
591 ˜˜˜ }
592 ˜˜˜ else if(actualCola == ultimoCola){
593      cout<< "(" << actualCola->x<<","<<actualCola->y<< ")";
594 ˜˜˜ }
595 ˜˜˜ actualCola = actualCola->siguienteCola;
596 ˜˜˜ }
597 ˜˜˜ cout<<"\n\n";
598 ˜˜˜ }else{
599 ˜˜˜ cout << endl << " La cola se encuentra Vacia " << endl << endl;
600 ˜˜˜ }
601 ˜˜˜ }
602 }
```

Se hace una impresión de la cola guardada anteriormente, indicando cual es la salida de los movimientos.

4.5. Ver Articulos de Tienda

```
1382 void Mostrar_Tienda(int monedas){
1383     NodoCategoria* aux1 = new NodoCategoria();
1384     aux1 = primeroCategoria;
1385
1386     NodoArticulos* aux2 = new NodoArticulos();
1387     cout<<"\t\tTotal Tokens: "<<monedas<<endl;
1388     cout<<"ID "<<" Nombre"<<"\tCategoria"<<" Precio"<<endl;
1389     while(aux1!=NULL){
1390         aux2 = aux1->abajo;
1391         while (aux2!=NULL)
1392         {
1393             cout<<aux2->idArticulo<<"\t" <<aux2->nombreArticulo<<"\t" <<aux1->categoria<<"\t" <<aux2->precioArticulo<<endl;
1394             aux2 = aux2->siguienteArtic;
1395         }
1396         aux1 = aux1->siguienteCA;
1397     }
1398 }
1399 }
```

Se hace un recorrido de las lista de listas para mostrar todos los elementos en columnas.

4.6. Realizar Movimientos

```
1072 void movimientos(string nombreuser){  
1073     int movx, movy;  
1074     int resp;  
1075     cout<<"Ingrese la coordenada X: ";  
1076     cin>>movx;  
1077     cout<<"Ingrese la coordenada Y: ";  
1078     cin>>movy;  
1079     insertarPila(movx,movy);  
1080  
1081     cout<<"\nMovimiento - "<<movx<<","<<movy<<endl;  
1082  
1083     cout<<"Desea realizar otro movimiento? "<<endl;  
1084     cout<<"1. SI"<<endl;  
1085     cout<<"2. NO"<<endl;  
1086     cin>>resp;
```

Se pide la coordenada X, luego la coordenada Y y esta es guardada en una pila de la forma (X,Y). Siguiente se le pregunta al ususario si desea hacer otro movimiento.

```
1087 do{  
1088     switch (resp)  
1089     {  
1090         case 1:  
1091             cout<<"Ingrese la coordenada X: ";  
1092             cin>>movx;  
1093             cout<<"Ingrese la coordenada Y: ";  
1094             cin>>movy;  
1095             insertarPila(movx,movy);  
1096             cout<<"Desea realizar otro movimiento? "<<endl;  
1097             cout<<"1. SI"<<endl;  
1098             cout<<"2. NO"<<endl;  
1099             cin>>resp;  
1100             if(resp==2){  
1101                 string nombremov;  
1102                 cout<<"Nombre para guardar movimientos: ";  
1103                 cin>>nombremov;  
1104                 nombrejugada = nombremov;  
1105                 cout<<"Se ha guardado la jugada\n";  
1106             }  
1107         break;
```

Al seleccionar si, se vuelve a repetir la peticion de coordenadas hasta que seleccione “NO”.

```

1108     case 2:
1109         string nombremov;
1110         cout<<"Nombre para guardar movimientos: ";
1111         cin>>nombremov;
1112         nombrejugada = nombremov;
1113         cout<<"Se ha guardado la jugada\n";
1114     }
1115 }while(resp!=2);
1116
1117 }

```

Cuando selecciona No, se pide un nombre para guardar el/los movimientos realizados, posteriormente para ser graficados.

5. Reportes

```

300 void reportes(){
301     int opreport;
302     cout<<"\n";
303     cout<<"***** Menu reportes *****"=>>endl;
304     cout<<"1. Estructuras Utilizadas"=>>endl;
305     cout<<"2. Listado de usuarios ordenados por edad"=>>endl;
306     cout<<"3. Listado de articulos ordenados por precio"=>>endl;
307     cout<<"4. Regresar al menu principal"=>>endl;
308     cout<<"*****"=>>endl;
309     cin >> opreport;
310     cout<<"\n";

```

Para la opcion reportes se tienen 3 opciones de todas las estructuras usadas en el programa.

5.1. Estructuras Utilizadas

```

313     case 1:
314         int opcestruct;
315         cout<<"1. Lista Usuarios"=>>endl;
316         cout<<"2. Lista Articulos"=>>endl;
317         cout<<"3. Tutorial"=>>endl;
318         cout<<"4. Listado de Jugadas"=>>endl;
319         cout<<"5. Regresar al menu principal"=>>endl;
320         cin>> opcestruct;
321         cout<<"\n";

```

Se puede graficar Lista Usuario, Lista Articulos, Tutorial o Listado de jugadas.

```
322     switch (opcstruct){  
323     case 1:  
324         GraficoListaCDobleEnlace();  
325         break;  
326     case 2:  
327         GraficoListadeListas();  
328         break;  
329     case 3:  
330         GraficoTutorial();  
331         break;  
332     case 4:  
333         GraficoMovimientos(userlogin,nombrejugada);  
334         break;  
335     case 5:  
336         cout<<"\n";  
337         return;  
338         break;  
339     default:  
340         cout<<"\nIngrese una opcion correcta\n"<<endl;  
341         break;  
342     }  
343     break;  
344 }
```

Se mandan a llamar a las funciones de Graficas.

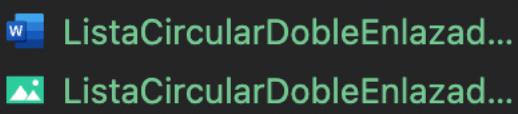
```
981 void GraficoListaCDobleEnlace(){  
982     nodo* actual = new nodo();  
983     actual = primero;  
984     int contador = 0;  
985     int contador2 = 0;  
986     string nombreNodo, direccion;  
987  
988     if(actual==NULL){  
989         cout<<"No se puede graficar porque no existen datos en la lista"<<endl;  
990     }else{  
991         string dot = "";  
992         dot = dot + "digraph G {\n";  
993         dot = dot + "graph [nodesep=\"0.75\"]\n";  
994         dot = dot + "labelloc=\"t\"\n";  
995         dot = dot + "label=\"Lista Circular doblemente enlazada\"\n";  
996         dot = dot + "node[shape=box]" + "\n";  
997         if (primero!=NULL) {  
998             do {  
999                 nombreNodo = "nodo"+to_string(contador);  
1000                dot = dot + nombreNodo + "[label =" + (actual->nombreuser) + "\nContra: " +(actual->contracifrada) + "\nMor  
1001                if(actual->siguiente!=primero){  
1002                    int auxnum = contador +1;  
1003                    int prueba = contador2 -1;  
1004                    direccion += nombreNodo + "-> nodo" + std::to_string(auxnum) + "[dir=both];\n";  
1005                }else{  
1006                    int auxnum = contador + 1;  
1007                    direccion += nombreNodo + ":" + "n" + "-> nodo" + std::to_string(0) + ";\n";  
1008                    direccion += "nodo" + std::to_string(0) + "s" + "->" + nombreNodo + ";\n";  
1009                }  
1010                actual = actual -> siguiente;  
1011                contador++;  
1012            } while(actual!=primero);  
1013        }  
1014        dot += nombreNodo + "\n";  
1015        dot += "rank=same;\n" + direccion + "\n";  
1016        dot = dot + "\n";
```

Este es la forma de generar un archivo .dot de graphviz de una lista circular doblemente enlazada, donde usamos estructura de graphviz sumado a que se le mandan valores que se necesitan, como el nick, contraseña, monedas y edad.

```

1018     ofstream file;
1019     file.open("ListaCircularDobleEnlazada.dot");
1020     file << dot;
1021     file.close();
1022     system(("dot -Tpng ListaCircularDobleEnlazada.dot -o ListaCircularDobleEnlazada.png"));
1023     cout<<"\nReporte Usuario Generado\n"<<endl;
1024 }
1025 }
```

Se crea el archivo .dot y luego se convierte a imagen con un comando del sistema.



La salida debe ser la siguiente, el archivo .dot y la imagen en formato .png porque esta extensión se le da.

Así sucesivamente con los demás grafos.

5.2. Lista de Usuarios Ordenadas

```

345     case 2:
346         int ordenl;
347         cout<<"1. Orden Ascendente" << endl;
348         cout<<"2. Orden Descendente" << endl;
349         cout<<"3. Regresar al menu principal" << endl;
350         cin >> ordenl;
351
352         switch (ordenl)
353         {
354             case 1:
355                 cout << "lista usuarios de forma ascendente" << endl;
356                 ordenarUsuarioASC();
357
358                 break;
359
360             case 2:
361                 cout << "lista usuarios de forma descendente" << endl;
362                 ordenarUsuarioDESC();
363
364                 break;
365
366             case 3:
367                 cout << "\n";
368                 return;
369                 break;
370 }
```

Se cuenta con forma Ascendente o Descendente.

```

1447 void ordenarUsuarioASC(){
1448     nodo* actual = new nodo();
1449     actual = primero;
1450     int auxi,auxi2;
1451
1452     if(primer!=NULL){
1453         do{
1454             nodo* aux = actual->siguiente;
1455             while(aux!=NULL){
1456                 if(actual->edad > aux->edad){
1457                     auxi = aux->edad;
1458                     aux->edad = actual->edad;
1459                     actual->edad = auxi;
1460                     cout<<"Edad1: "<<auxi<<endl;
1461                 }else{
1462                     auxi2 = aux->edad;
1463                     aux->edad = actual->edad;
1464                     aux = actual->siguiente;
1465                     actual->edad = auxi2;
1466                     cout<<"Edad2: "<<auxi2<<endl;
1467                 }
1468                 aux = aux->siguiente;
1469                 actual = actual->siguiente;
1470             }
1471         }while(actual->siguiente!=primero);
1472     }
1473 }
```

Ordenado por el método Bubble Sort

Para el descendente se realizan los mismos pasos, únicamente cambiado la condición para “<”.

5.3. Lista de Articulos Ordenadas

```

373 case 3:
374     int ordenp;
375     cout<<"1. Orden Ascendente"<<endl;
376     cout<<"2. Orden Descendente"<<endl;
377     cout<<"3. Regresar al menu principal"<<endl;
378     cin>> ordenp;
379
380     switch (ordenp)
381     {
382     case 1:
383         cout<<"lista articulos de forma precio ascendente"<<endl;
384         ordenarPrecioASC();
385         break;
386
387     case 2:
388         cout<<"lista usuarios de forma precio descendente"<<endl;
389         ordenarPrecioDESC();
390         break;
391     case 3:
392         cout<<"\n";
393         return;
394         break;
395     default:
396         cout<<"Ingrese una opcion correcta\n";
397         break;
398     }
```

Se cuenta con forma Ascendente o Descendente.

```
1396 void ordenarPrecioASC(){
1397
1398     NodoArticulos *nodo1 = new NodoArticulos();
1399     NodoArticulos *actual, *siguiente;
1400     int n;
1401     if(nodo1 != NULL)
1402     {
1403         actual = nodo1;
1404         do
1405         {
1406             siguiente = actual->siguienteArtic;
1407             while(siguiente != nodo1)
1408             {
1409                 if(actual->precioArticulo > siguiente->precioArticulo)
1410                 {
1411                     n = siguiente->precioArticulo;
1412                     siguiente->precioArticulo = actual->precioArticulo;
1413                     actual->precioArticulo = n;
1414                 }
1415                 siguiente = siguiente->siguienteArtic;
1416             }
1417             actual = actual->siguienteArtic;
1418             siguiente = actual->siguienteArtic;
1419         }
1420         while(siguiente != nodo1);
1421     }
1422 }
```

Ordenado por el método Bubble Sort

Para el descendente se realizan los mismos pasos, únicamente cambiado la condición para “<”.