

Escuela de Ciencias y Sistemas

Estructura de Datos

Proyecto – Fase 3
Manual Técnico

César André Ramírez Dávila

202010816

Fecha: 27/10/2022

Índice

<i>Introducción</i>	4
<i>Objetivos</i>	4
<i>Requisitos.....</i>	5
<i>Estructuras de la Fase 1</i>	6
<i>Creación del programa</i>	6
1. Librerías	6
1.1. Archivos de cabecera	6
1.2. Archivo de entrada.....	6
2. <i>Main.cpp</i>	8
2.1. Funciones declaradas	8
2.2. Nodos	8
2.3. Menú principal.....	9
2.4. Carga Masiva.....	9
2.5. Registros JSON	11
3. <i>Registro Usuarios.....</i>	14
4. <i>Login.....</i>	14
4.1. Menu Login	15
4.2. Editar información	15
4.3. Eliminar Cuenta.....	17
4.4. Ver Tutorial	18
4.5. Ver Articulos de Tienda	18
4.6. Realizar Movimientos.....	19
5. <i>Reportes</i>	20
5.1. Estructuras Utilizadas.....	20
5.2. Lista de Usuarios Ordenadas	23
5.3. Lista de Articulos Ordenadas	24
<i>Estructuras de la Fase 2</i>	25
<i>Creación del programa</i>	25
1. Librerías	25
1.1. Archivos de Cabecera	25
1.2. Archivo de Entrada.....	26
2. Servidor	27

3.	Main.cpp.....	30
3.1.	Conexión Servidor	31
4.	Cliente.py.....	34
4.1.	Registro Usuarios	35
4.2.	Login	36
4.3.	Administrador	38
4.4.	Usuarios.....	39
5.	Tablero de Juego	42
6.	Salida	44
	<i>Estructuras de la Fase 3</i>	<i>44</i>
	<i>Creación del programa</i>	<i>44</i>
1.	Jugador vs Jugador	44
1.1.	Tablero.....	45
1.2.	Lista de Adyacencia	50
1.3.	Grafo de la lista	52

Introducción

El presente documento describe los aspectos técnicos informáticos del programa. El documento familiariza al desarrollador que hace uso del lenguaje C++, Python y una conexión entre estos 2 lenguajes, con temas como: uso de nodos, listas, ordenamientos, ciclos repetitivos, cargas de archivos con muchos datos, registros, graficas, crear tablero de juegos, jugadas, seguridad en compras ,entre otros.

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++
- Comprender uso de Interfaces graficas en el lenguaje de Python
- Obtener mayor conocimiento en el lenguaje C++ y Python
- Aprender el uso de seguridad informatica.

El presente manual está enfocado en la comunicacion entre lenguaje de programación C++ combinado con Python.

Requisitos

- La aplicación puede ser ejecutada en distribuciones de Linux y MacOS y Windows, únicamente debe tener 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 4gb

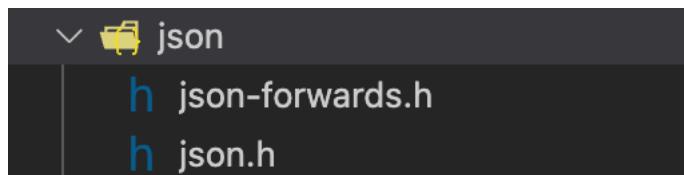
Estructuras de la Fase 1

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": "5", "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_usuario(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 GraficolistadoDobleEnlace();
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           *"
125         cout<<"* 2. Registrar Usuario      *"
126         cout<<"* 3. Login                  *"
127         cout<<"* 4. Reportes               *"
128         cout<<"* 5. Salir del juego        *"
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,srcarticulo;
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         precioarticuloo = 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 = precioarticuloo;
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_usuario(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                 primer -> siguiente=primer;
440                 primer -> anterior=ultimo;
441             }else{
442                 ultimo-> siguiente=nuevo;
443                 nuevo-> anterior=ultimo;
444                 nuevo-> siguiente=primer;
445                 ultimo=nuevo;
446                 primer-> 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(primer == 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 }
1248

```

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(primerоЮla==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 encriptado = SHA256::cifrar(contra);
258         //cout<<"El cifrado sha es : "<<encriptado<<endl;
259         registro_usuario(nombreuser,contra,monedas, edad,encriptado);
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
295         }else{
296             cout << "\nNo existe el usuario en la lista\n\n";
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 Información      *" <<endl;  
701         cout<<"* 2. Eliminar Cuenta        *" <<endl;  
702         cout<<"* 3. Ver tutorial           *" <<endl;  
703         cout<<"* 4. Ver artículos 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 deseaa 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!=primer && 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 búsqueda de la posición 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 búsqueda de la posición 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(primerоЮCola!=NULL){
584         cout<<"\tAncho: " << primeroCola->x << "\n";
585         cout<<"\tAlto: " << primeroCola->y << "\n";
586         cout<<"    Movimientos: " << endl;
587         cout<<"\t";
588     ~~~
589     ~~~
590     ~~~
591     ~~~
592     ~~~
593     ~~~
594     ~~~
595     ~~~
596     ~~~
597     ~~~
598     ~~~
599     }else{
600         cout << endl << " La cola se encuentra Vacia " << endl << endl;
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\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 << " " << 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 petición 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 opción 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 (opcestruct){  
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;
```

Se mandan a llamar a las funciones de Graficas.

```

981 void GraficoListaDobleEnlace(){
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=" + Nick + " " + (actual->nombreuser) + "\nContra: " + (actual->contracifrada) + "\nMonedas: " + (actual->monedas) + "\nEdad: " + (actual->edad) + "]";
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";

```

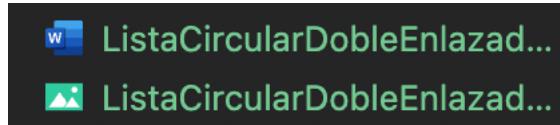
Este es la forma de generar un archivo .dot de graphviz de una lista circular doblemente enlazada, donde usamos estructura de graphviz sumado a que se le mandan valores que se necesitan, como el nick, contraseña, monedas y edad.

```

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;
```

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         }
1421     }
```

Ordenado por el método Bubble Sort

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

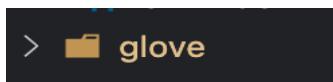
Estructuras de la Fase 2

Creación del programa

1. Librerías

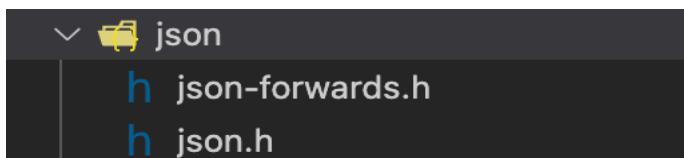
Para poder realizar la comulación entre C++ y Python se usa la librería glove, esta debe ser clonada en nuestra carpeta del proyecto.

Enlace: <https://github.com/gasparfm/glove>



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



1.1. Archivos de Cabecera

```

#include "./glove/glovehttpserver.hpp"
#include <iostream>
#include <chrono>
#include <thread>
#include <string>
#include <vector>
#include <fstream>
#include "./glove/json.hpp"
#include "C++/jsoncpp.cpp"
#include "C++/ListaCircular.cpp"
#include "C++/ListaArticulos.cpp"
#include "C++/ListaTutorial.cpp"
#include "C++/ArbolB.cpp"
#include "C++/ListaPila.cpp"

```

1.2. Archivo de Entrada

```

{
    "usuarios": [
        { "id": "1", "nick": "andre", "password": "card", "monedas": "200", "edad": "20" },
        { "id": "2", "nick": "luis", "password": "elpana", "monedas": "100", "edad": "40" },
        { "id": "3", "nick": "angel", "password": "siquea", "monedas": "50", "edad": "21" },
        { "id": "4", "nick": "pedro", "password": "pedrovml", "monedas": "350", "edad": "10" }
    ],
}

```

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

```

"articulos": [
    { "id": "1", "categoria": "legendario", "precio": "500", "nombre": "barco1", "src": "link1" },
    { "id": "2", "categoria": "epic", "precio": "400", "nombre": "barco2", "src": "link2" },
    { "id": "3", "categoria": "basico", "precio": "100", "nombre": "barco3", "src": "link3" },
    { "id": "4", "categoria": "basico", "precio": "300", "nombre": "barco4", "src": "link4" }
],
}

```

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

```

"tutorial": {
    "ancho": "20",
    "alto": "30",
    "movimientos": [
        { "x": "1", "y": "6" },
        { "x": "3", "y": "9" },
        { "x": "5", "y": "1" },
        { "x": "8", "y": "3" },
        { "x": "1", "y": "0" }
    ]
}

```

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

2. Servidor

Para realizar la conexión de los lenguajes, es necesario tener archivos header “.h” para la correcta comunicación con glove.

C++ ArbolB.cpp

h ArbolB.h

Ejemplo de archivo .h

Nodo para los usuarios

```
#ifndef NODOUSUARIO_H
#define NODOUSUARIO_H
#include <stddef.h>
#include <string>

class nodoUsuarios{
public:
    std::string nombreuser;
    std::string contra;
    std::string contracifrada;
    int id,monedas,edad;
    nodoUsuarios *anterior;
    nodoUsuarios *siguiente;
    nodoUsuarios(){
        anterior = NULL;
        siguiente = NULL;
        id = 0;
        nombreuser = ' ';
        contra = ' ';
        contracifrada = ' ';
        monedas = 0;
        edad = 0;
    }
private:
};
#endif
```

Creando la lista para usuarios

```
#ifndef LISTACIRCULAR_H
#define LISTACIRCULAR_H

#include "NodoUsuarios.h"
#include <iostream>
using namespace std;

class ListaCircular{
public:
    nodoUsuarios*primero;
    nodoUsuarios*ultimo;
    ListaCircular(){
        primero = NULL;
        ultimo = NULL;
    }
}
```

Métodos que tendrá la lista

```
void registro_usuario(int id,string nombreuser, string contracifrada, int monedas ,int edad, string contra);
void registro_usuarioJ(int id,string nombreuser, string contracifrada, int monedas ,int edad, string contra);
void lista_usuarios();
void ordenarUsuarioASC();
void ordenarUsuarioDESC();
void ListaUsuarioASC(nodoUsuarios *cabeza);
void ListaUsuarioDESC(nodoUsuarios *cabeza);
void intercambioASC(nodoUsuarios *lado_izq, nodoUsuarios *lado_der);
void intercambioDESC(nodoUsuarios *lado_izq, nodoUsuarios *lado_der);
void HtmlASC(nodoUsuarios *cabeza);
void HtmlDESC(nodoUsuarios *cabeza);
void eliminarCuenta(string userbuscado);
void modificarUsuario(string userb, string nuevouser, string contra, int edad, string cifrada);
string BuscarUser(string nombreuser);
string getUsers();
string Comprobar(string nombreuser);
string Comprobar1(string nombreuser);
string verificarLog(string usuario, string cifrada);
string Buscar1(string username);
```

```
private:
};

#endif
```

Los métodos para agregar usuarios son los mismos descritos en el manual de la Fase 1

Los nuevos son: getUsers()

```
string ListaCircular::getUsers() {
    nodoUsuarios*aux = primero;
    string datos = "";
    datos += "{\"usuario\":[";

    while (aux != NULL){
        datos += "{";
        datos+= "\"id\":\"" + to_string(aux->id) + "\",";
        datos+= "\"nick\":\"" + aux->nombreuser + "\",";
        datos+= "\"password\":\"" + aux->contracifrada + "\",";
        datos+= "\"monedas\":\"" + to_string(aux->monedas) + "\",";
        datos+= "\"edad\":\"" + to_string(aux->edad) + "\"";
        datos += "}";

        aux = aux->siguiente;
        if(aux!=primero){
            datos+= ",";
        }
        if(aux == primero){
            break;
        }
    }

    datos += "]";
    return datos;
}
```

Creamos un string con formato json que nos servirá para trabajar con los datos en Python más adelante.

Así sucesivamente con todas las estructuras de datos que se utilizan en el proyecto.

3. Main.cpp

Este archivo será la fuente de comunicación con Python.

```
int contadorusuarios = 0;
ListaCircular ListaUsuarios;
ListaArticulos ListaArt;
ListaTutorial ListTutorial;
ArbolB Arbol;
ListaPila Movimientos;
```

Se define un contador para los usuarios y las listas para almacenar datos creadas previamente.

```
int atoi(std::string s)
{
    try
    {
        return std::stod(s);
    }
    catch (std::exception &e)
    {
        return 0;
    }
}
```

El método llamado atoi nos permite pasar datos de tipo string a formato int.

```
static std::string jsonkv(std::string k, std::string v)
{
    /* "k": "v" */
    return "\"" + k + "\": \"" + v + "\"";
}
```

Esto nos permite retornar una respuesta en formato json.

3.1. Conexión Servidor

```
class Servidor
{
public:
    Servidor()
    {
        string usuario = "EDD";
        string contra = "edd123";
        string encriptado = SHA256::cifrar(contra);
        ///cout<<"encriptado: "<<encriptado<<"\n";

        ListaUsuarios.registro_usuario(contadorusuarios,usuario,encriptado,0,50,contra);
        Arbol.insertar(contadorusuarios,usuario);
        contadorusuarios++;
    }
}
```

Al iniciar el servidor se agrega el usuario administrador.

```
void get(GloveHttpRequest &request, GloveHttpResponse &response)
{
    string rut;
    response.contentType("text/json");
    rut = request.special["Ruta"];
    if (rut.empty())
        response << ListaUsuarios.getUsers();
    else
    {
        response << "{ "
        << jsonkv("status", "ok tengo el get") << ",\n"
        " }";
    }
}
```

Se crea un método get para obtener la ruta del archivo a cargar.

```

void post(GloveHttpRequest &request, GloveHttpResponse &response)
{
    ifstream archivo;
    string ruta;
    string pruebaa;
    string texto;
    string iduser,nombreuser,contra,monedas,edad;
    string idarticoloo, categoriarticulo,precioarticoloo,nombrearticulo,srcarticulo;
    string alt, anch, x1, y1;
    ruta = request.special["Ruta"];
    archivo.open(ruta.c_str(), ios::in);
    if(archivo.fail()){
        cout<<"\nNo se pudo abrir el archivo\n"<<endl;
    }
}

```

Hacemos un método post para la lectura del archivo, similar a la de la fase 1.

```

while (!archivo.eof())
{
    Json::Reader reader;
    Json::Value obj;
    reader.parse(archivo, obj);
    const Json::Value& usuariosJ = obj["usuarios"];
    for (int i = 0; i < usuariosJ.size(); i++){
        iduser = usuariosJ[i]["id"].asString();
        //cout << "\nNick: " << usuariosJ[i]["nick"].asString();
        nombreuser = usuariosJ[i]["nick"].asString();
        //cout << "\nPass: " << usuariosJ[i]["password"].asString();
        contra = usuariosJ[i]["password"].asString();
        string encriptado = SHA256::cifrar(contra);
        //cout<<"El cifrado sha es : "<<encriptado<<endl;
        //cout << "\nMonedas: " << usuariosJ[i]["monedas"].asString();
        monedas = usuariosJ[i]["monedas"].asString();
        //cout << "\nEdad: " << usuariosJ[i]["edad"].asString();
        edad = usuariosJ[i]["edad"].asString();
        std ::string idi = iduser;
        std ::string edadi = edad;
        std ::string monedasi = monedas;
        int iddi = std::stoi(idi);
        int eddi = std::stoi(edadi);
        int monedi = std::stoi(monedasi);
        ListaUsuarios.registro_usuarioJ(iddi,nombreuser, encriptado, monedi, eddi, contra);
    }
}

```

Así sucesivamente con los artículos y el tutorial del Juego.

```

const Json::Value& articulosJ = obj["articulos"];
for (int i = 0; i < articulosJ.size(); i++){
    //cout << "\nID: " << articulosJ[i]["id"].asString();
    idarticuloo = articulosJ[i]["id"].asString();
    //cout << "\nCategoria: " << articulosJ[i]["categoria"].asString();
    categoriarticulo = articulosJ[i]["categoria"].asString();
    //cout << "\nPrecio: " << articulosJ[i]["precio"].asString();
    precioarticulo = articulosJ[i]["precio"].asString();
    //cout << "\nNombre: " << articulosJ[i]["nombre"].asString();
    nombrearticulo = articulosJ[i]["nombre"].asString();
    //cout << "\nSRC: " << articulosJ[i]["src"].asString();
    srcarticulo = articulosJ[i]["src"].asString();
    std ::string iarticulo = idarticuloo;
    std ::string precioarticul = precioarticulo;
    int precioarticulo = std::stoi(precioarticul);
    ListaArt.registro_articulos(categoriarticulo,nombrearticulo,precioarticulo,idarticuloo,srcarticulo);
    //cout << endl;
}

```

```

const Json::Value& tutorialJ = obj["tutorial"];
//cout <<"\nAncho: "<<tutorialJ["ancho"].asString();
anch = tutorialJ["ancho"].asString();
//cout<<"\nAlto: "<<tutorialJ["alto"].asString();
alt = tutorialJ["alto"].asString();
std ::string ancho = anch;
std ::string alto = alt;
int x = std::stoi(ancho);
int y = std::stoi(alto);
ListTutorial.registroTutorial(x,y);
//cout<<"\nMovimientos: ";
const Json::Value& movimientosJ = tutorialJ["movimientos"];
for(int i = 0; i < movimientosJ.size(); i++){
    //cout << "\nX: " << movimientosJ[i]["x"].asString();
    x1 = movimientosJ[i]["x"].asString();
    //cout << " Y: " << movimientosJ[i]["y"].asString();
    y1 = movimientosJ[i]["y"].asString();
    int x = std::stoi(x1);
    int y = std::stoi(y1);
    ListTutorial.registroTutorial(x,y);
}
cout<<"\n";
cout<<"\nArchivo cargado con exito\n"<<endl;
break;
}
archivo.close();

```

```

response << "{ "
    << jsonkv("status", "ok ha sido enviado") << ",\n"
    " }";
return;
}

```

Si el intercambio de datos es correcto, retornamos un mensaje de éxito.

4. Cliente.py

```

ventana = Tk()
ventana.title("Proyecto Fase 2 - 202010816")
ventana.resizable(0,0)
ancho_ventana = 500
alto_ventana = 500
x_ventana = ventana.winfo_screenwidth() // 2 - ancho_ventana // 2
y_ventana = ventana.winfo_screenheight() // 2 - alto_ventana // 2
posicion = str(ancho_ventana) + "x" + str(alto_ventana) + "+" + str(x_ventana) + "+" + str(y_ventana)
ventana.geometry(posicion)
#Botones
btnCargarArchivo = Button(ventana, height=2, width=15, text="Cargar Usuarios", command = abrirArchivo1, background="#B03314", fg="white")
btnCargarArchivo.place(x=180, y=150)
btnRegistrar = Button(ventana, height=2, width=15, text="Registrar Usuario", command=lambda:[ ventana.withdraw(),ventanaLog.deiconify()])
btnRegistrar.place(x=180, y=210)
btnLogin = Button(ventana, height=2, width=15, text="Login",command=lambda:[ventana.withdraw(),ventanaLog.deiconify()])
btnLogin.place(x=180, y=270)
btnSalir = Button(ventana, height=2, width=15, text="Salir",command=cerrar, background="#B03314", font=("Verdana",10))
btnSalir.place(x=180, y=330)
#Labels
labelEditor = Label (ventana, text ="BATALLA NAVAL", font=("Verdana",16), background="#044D9A", fg="white")
labelEditor.place(x=180, y=90)

```

Empezamos con la interfaz gráfica en Python, luego llamamos a la función que tendrá el servidor para el intercambio de datos.

```

base_url = "http://127.0.0.1:8080/"

def abrirArchivo1():
    try:
        global archivo
        archivo = filedialog.askopenfilename(title="Seleccionar archivo", filetypes=[("json", "*.json")])
        #print(archivo)
        prueba = os.path.split(archivo)
        #print(prueba)
        res = requests.post(f'{base_url}/Carga/' + f'{prueba[1]}')
        data = res.text#convertimos la respuesta en dict
        print(data)
        MessageBox.showinfo("Exito!", "Archivo Cargado con exito")
    except:
        MessageBox.showwarning("Alerta", "Debe cargar un archivo")

```

Se tiene la url de conexión y se hace la petición al servidor.

4.1. Registro Usuarios

```
ventanaReg = Toplevel()
ventanaReg.title("Registro Usuario")
ventanaReg.resizable(0,0)
ancho_ventana1 = 500
alto_ventana1 = 500
x_ventana1 = ventanaReg.winfo_screenwidth() // 2 - ancho_ventana1 // 2
y_ventana1 = ventanaReg.winfo_screenheight() // 2 - alto_ventana1 // 2
posicion1 = str(ancho_ventana1) + "x" + str(alto_ventana1) + "+" + str(x_ventana1) + "+" + str(y_ventana1)
ventanaReg.geometry(posicion1)
labelLogin = Label (ventanaReg, text ="Registro", font=("Verdana",16), background="#044D9A", fg="white")
labelLogin.place(x=210, y=80)
labelUser = Label (ventanaReg, text ="Ingrese Usuario", font=("Verdana",16), background="#044D9A", fg="white")
labelUser.place(x=50, y=200)
labelPass = Label (ventanaReg, text ="Ingrese Contraseña", font=("Verdana",16), background="#044D9A", fg="white")
labelPass.place(x=50, y=260)
labelEdad = Label (ventanaReg, text ="Ingrese Edad", font=("Verdana",16), background="#044D9A", fg="white")
labelEdad.place(x=50, y=320)
textoUsuario = Text(ventanaReg, height=2, width=30, fg="white", font=("Consolas", 11))
textoUsuario.place(x=230, y=190)
textoPass = Text(ventanaReg, height=2, width=30, fg="white", font=("Consolas", 11))
textoPass.place(x=230, y=255)
textoEdad = Text(ventanaReg, height=2, width=30, fg="white", font=("Consolas", 11))
textoEdad.place(x=230,y=330)
btnRegistro = Button(ventanaReg, height=2, width=15, text="Registrar", command = lambda:[Comprobar(textoUsuario.get(1,END), textoPass.get(1,END), textoEdad.get(1,END))])
btnRegistro.place(x=180, y=400)
btnRegreso = Button(ventanaReg, height=2, width=8, text="Regresar", command = lambda:[textoUsuario.delete(1.0, tk.END), textoPass.delete(1.0, tk.END), textoEdad.delete(1.0, tk.END)])
btnRegreso.place(x=405, y=5)
ventanaReg.withdraw()
```

Se crea una ventana y luego se llama a la función que nos registrará el usuario en C++.

```
def Comprobar(salida,salida2,salida3):
    res = requests.get(f'{base_url}/Verificar/' + f'{salida}' + "/" + f'{salida2}' + "/" + f'{salida3}')
    data = res.text#convertimos la respuesta en dict

    datos_diccionarioo = json.loads(data)
    #idd1 = datos_diccionario2["Id"]
    estt = datos_diccionarioo["estado"]
    #print(data)

    if estt == "existe":
        MessageBox.showinfo("Problema", "Ya existe un usuario con este Nick")
    if estt == "no existe":
        mandarRegistro(salida,salida2,salida3)
        textoUsuario.delete(1.0, tk.END+"-1c")
        textoPass.delete(1.0, tk.END+"-1c")
        textoEdad.delete(1.0, tk.END+"-1c")
        ventanaReg.withdraw()
```

Haciendo una petición donde verifica si el usuario a ingresar ya existe o no, en caso de que no exista no nos permite crearlo y nos muestra una advertencia, de lo contrario hace el registro de forma exitosa.

```

def mandarRegistro(salida,salida2,salida3):
    res = requests.get(f'{base_url}/Registro/' + f'{salida}' + "/" + f'{salida2}' + "/" + f'{salida3}')
    data = res.text#convertimos la respuesta en dict
    cerrandoRegistro()

def cerrandoRegistro():
    #print("llegando")
    MessageBox.showinfo("Exito!", "Usuario Registrado con exito")
    ventana.deiconify()

```

Llamando a otras funciones para completar el registro.

4.2. Login

```

ventanaLog = Toplevel()
ventanaLog.title("Login")
ventanaLog.resizable(0,0)
ancho_ventana1 = 500
alto_ventana1 = 500
x_ventana1 = ventanaLog.winfo_screenwidth() // 2 - ancho_ventana1 // 2
y_ventana1 = ventanaLog.winfo_screenheight() // 2 - alto_ventana1 // 2
posicion1 = str(ancho_ventana1) + "x" + str(alto_ventana1) + "+" + str(x_ventana1) + "+" + str(y_ventana1)
ventanaLog.geometry(posicion1)
bgL = PhotoImage(file="Python/user.png")
labelPhotoL = Label(ventanaLog, image=bgL)
labelPhotoL.place(x=150,y=30)
labelLogin = Label (ventanaLog, text ="Login", font=("Verdana",16), background="#044D9A", fg="white")
labelLogin.place(x=230, y=200)
labelUser = Label (ventanaLog, text ="Ingrese Usuario", font=("Verdana",16), background="#044D9A", fg="white")
labelUser.place(x=40, y=280)
labelPass = Label (ventanaLog, text ="Ingrese Contraseña", font=("Verdana",16), background="#044D9A", fg="white")
labelPass.place(x=40, y=340)
textoUsuarioL = Text(ventanaLog, height=2, width=30, fg="white", font=("Consolas", 12))
textoUsuarioL.place(x=220, y=270)
textoPassL = Entry(ventanaLog, fg="white", show="*", font=("Consolas", 12), width=30)
textoPassL.place(x=220, y=340)
btnIniciar = Button(ventanaLog, height=2, width=15, text="Iniciar Sesion", command = lambda:[mandarLogin(textoUsuarioL.get(), textoPassL.get())])
btnIniciar.place(x=110, y=410)
btnRegresar = Button(ventanaLog, height=2, width=15, text="Regresar", command = lambda:[ventana.deiconify()])
ventanaLog.withdraw()

```

Creamos la ventana y por cedemos a llamar a la función que nos verifica los datos del login.

```

def mandarLogin(usuario, contra):
    global usuariobusqueda, monedasusuario
    print(usuario)
    print(contra)
    res = requests.get(f'{base_url}/Login/' + f'{usuario}' + "/" + f'{contra}')
    data = res.text#convertimos la respuesta en dict
    print(data)
    if data == "admin":
        ventanaLog.withdraw()
        MessageBox.showinfo("Exito!", "Inicio de sesion correcto")
        ventanaAdmin.deiconify()
    if data == "correcto":
        ventanaLog.withdraw()
        MessageBox.showinfo("Exito!", "Inicio de sesion correcto")
        res = requests.get(f'{base_url}/Log/' + f'{usuario}' + "/" + f'{contra}')
        data = res.text
        datos_diccionario = json.loads(data)
        name = datos_diccionario["nick"]
        passw = datos_diccionario["password"]
        edadd = datos_diccionario["edad"]
        monedd = datos_diccionario["monedas"]
        textoUsuarioC.insert(INSERT, name)
        textoPassC.insert(INSERT, passw)
        textoEdadC.insert(INSERT, edadd)
        usuariobusqueda = name
        monedasusuario = int(monedd)
        ventanaUser.deiconify()
    if data == "incorrecto":
        MessageBox.showinfo("Error!", "Usuario o contraseña incorrectos")
        ventanaLog.deiconify()
    if data == "inexistente":
        MessageBox.showinfo("Inesperado!", "El usuario no existe")
        ventanaLog.deiconify()

```

Hacemos una petición al servidor donde mandamos los datos y comprobamos que existan en la lista guardada.

En este caso como hay un usuario administrador nuestro servidor debe verificar si este es el administrador o un usuario normal, en caso sea administrador abre la ventana para este, sino abre la ventana para usuarios normales.

4.3. Administrador

```
ventanaAdmin = Toplevel()
ventanaAdmin.title("ADMINISTRADOR")
ventanaAdmin.resizable(0,0)
ancho_ventana1 = 500
alto_ventana1 = 500
x_ventana1 = ventanaAdmin.winfo_screenwidth() // 2 - ancho_ventana1 // 2
y_ventana1 = ventanaAdmin.winfo_screenheight() // 2 - alto_ventana1 // 2
posicion1 = str(ancho_ventana1) + "x" + str(alto_ventana1) + "+" + str(x_ventana1) + "+" + str(y_ventana1)
ventanaAdmin.geometry(posicion1)
bg = PhotoImage(file="Python/admin.png")
labelPhoto = Label(ventanaAdmin, image=bg)
labelPhoto.place(x=150,y=30)
btnUser = Button(ventanaAdmin, height=2, width=22, text="Lista de usuarios", command = lambda: [verusuario()], background="white", foreground="black")
btnUser.place(x=160, y=200)
btnArticulos = Button(ventanaAdmin, height=2, width=22, text="Lista de articulos", command = lambda: [verArticulos()], background="white", foreground="black")
btnArticulos.place(x=160, y=250)
btnOrdenAsc = Button(ventanaAdmin, height=2, width=22, text="Usuarios ordenados ascendente", command = lambda: [verusuariosordenadoASC()], background="white", foreground="black")
btnOrdenAsc.place(x=160, y=300)
btnOrdenDesc = Button(ventanaAdmin, height=2, width=22, text="Usuarios ordenados descendente", command = lambda: [verusuariosordenadoDESC()], background="white", foreground="black")
btnOrdenDesc.place(x=160, y=350)
btnTutorial = Button(ventanaAdmin, height=2, width=22, text="Tutorial del juego", command = lambda: [Tutorial()], background="white", foreground="black")
btnTutorial.place(x=160, y=400)
btnCerrarSesionAdmin = Button(ventanaAdmin, height=2, width=22, text="Cerrar Sesion", command = lambda: [ventanaLog.delete()])
btnCerrarSesionAdmin.place(x=160, y=450)
ventanaAdmin.withdraw()
```

Creamos la ventana, pero en este caso el administrador tiene la posibilidad de acceder a todos los reportes, entonces cada función es una petición al servidor.

```
def verusuario():
    res = requests.get(f'{base_url}/Usuarios/')
    data = res.text#convertimos la respuesta en dict
    #print(data)
    im = Image.open('Arbol.png')
    im.show()
```

Para ver a los usuarios en el Árbol B, se hace una petición que devuelve una imagen con el Árbol y se procede a abrir automáticamente.

```
def verusuarioASC():
    res = requests.get(f'{base_url}/UsuariosASC/')
    data = res.text#convertimos la respuesta en dict
    #print(data)
```

4.4. Usuarios

```
ventanaUser = Toplevel()
ventanaUser.title("Principal")
ventanaUser.resizable(0,0)
ancho_ventana1 = 500
alto_ventana1 = 500
x_ventana1 = ventanaUser.winfo_screenwidth() // 2 - ancho_ventana1 // 2
y_ventana1 = ventanaUser.winfo_screenheight() // 2 - alto_ventana1 // 2
posicion1 = str(ancho_ventana1) + "x" + str(alto_ventana1) + "+" + str(x_ventana1) + "+" + str(y_ventana1)
ventanaUser.geometry(posicion1)
bgUser = PhotoImage(file="Python/usuario.png")
labelPhotoUser = Label(ventanaUser, image=bgUser)
labelPhotoUser.place(x=150,y=30)
btnEditar = Button(ventanaUser, height=2, width=15, text="Editar Informacion", command = lambda: [ventanaUser.withdraw(), EliminarCuenta()])
btnEditar.place(x=180, y=200)
btnEliminarCuenta = Button(ventanaUser, height=2, width=15, text="Eliminar mi cuenta", command = lambda: [EliminarCuenta()])
btnEliminarCuenta.place(x=180, y=250)
btnVerTutorial = Button(ventanaUser, height=2, width=15, text="Mostrar Tutorial", command = lambda: [Tutorial()])
btnVerTutorial.place(x=180, y=300)
btnVerTienda = Button(ventanaUser, height=2, width=15, text="Tienda", command = lambda: [MostrarTienda()])
btnVerTienda.place(x=180, y=350)
btnPartida = Button(ventanaUser, height=2, width=15, text="Iniciar Partida", command = lambda: [ventanaObtenerDimensiones()])
btnPartida.place(x=180, y=400)
btncerrarSesionUser = Button(ventanaUser, height=2, width=15, text="Cerrar Sesion", command = lambda: [ventanaUser.withdraw()])
btncerrarSesionUser.place(x=180, y=450)
ventanaUser.withdraw()
```

Creamos la ventana para usuarios normales, con sus distintas opciones, entre ellas la de jugar una partida.

```
def EliminarCuenta():
    res = requests.get(f'{base_url}/Eliminar/' + f'{usuariobusqueda}')
    data = res.text

    #print(data)
    datos_diccionario1 = json.loads(data)
    idd = datos_diccionario1["Id"]
    est = datos_diccionario1["estado"]
    if est == "encontrado":
        res = requests.get(f'{base_url}/Eliminando/' + f'{usuariobusqueda}' + "/" + f'{idd}')
        data = res.text
        mes = MessageBox.askquestion('Eliminar cuenta', '¿Esta seguro de eliminar esta cuenta?')
        if mes == 'yes':
            MessageBox.showinfo('Cerrando Sesion', 'La cuenta ha sido eliminada')
            ventanaUser.withdraw()
            ventanaLog.deiconify()
        else:
            MessageBox.showinfo('Regresar', 'Regresando al menu')
```

Para eliminar la cuenta, se le hace una advertencia al usuario de estar seguro de eliminarla.

```

def MostrarTienda():
    #print("Tienda")
    print(monedasusuario)
    global ides
    res = requests.get(f'{base_url}/Tienda/')
    data = res.text#convertimos la respuesta en dict
    #print(data)
    articulos = json.loads(data)

    idAr = []
    nombresAr = []
    cateAr =[]
    precioAr = []
    for articulo in articulos:
        ides = articulo.get('Id')
        nombr = articulo.get('nombre')
        catt = articulo.get('categoria')
        prec = articulo.get('precio')
        #print(articulo.get('Id'))
        #print(articulo.get('categoria'))
        #print(articulo.get('precio'))
        #print(articulo.get('nombre'))

        idAr.append(ides)
        nombresAr.append(nombr)
        cateAr.append(catt)
        precioAr.append(prec)

    fig = go.Figure(data=[go.Table(
        header=dict(values=['ID','Nombre','Categoria','Precio']),
        cells=dict(values=[idAr,nombresAr,cateAr,precioAr
                           ]))
    ])
    fig.update_layout(title = "Tienda", title_x=0.5)
    fig.show()

```

Mostrado la tienda

```

def Partida():
    global Portaaviones, Submarino, Destructores, Buques
    Portaav = 1
    Subma = 2
    Destruc = 3
    Buq = 4
    # Formula para determinar la cantidad de barcos por tablero
    # B(m) = ((m-1)/10)+1
    dimens = textoDimension.get(1.0, tk.END+"-1c")
    dimen = int(dimens)

```

Antes de empezar la partida se debe crear el tablero de juego, para ello obtenemos las dimensiones para este.

```

if dimen<10:
    MessageBox.showerror("Advertencia", "El Numero minimo para el tablero es de 10")
    if dimen > 10:

```

Si la dimensión es menor a 10 no se permite crear.

```

if dimen==10:
    ventanaObtenerDimension.withdraw()
    textoDimension.delete(1.0, tk.END+"-1c")
    vidas = 3
    B = int((dimen-1)/10)+1
    #print(B)
    Portaaviones = Portaav * B
    Submarino = Subma * B
    Destructores = Destruc * B
    Buques = Buq * B
    print("Portaaviones: ", Portaaviones)
    print("Submarinos : ", Submarino)
    print("Destructores: ", Destructores)
    print("Buques : " , Buques)
    MessageBox.showinfo("Exito", "Tablero creado con exito")
    Llamado(dimen,Buques,monedasusuario,vidas)

```

Si la dimensión es permitida, procede a crear el tablero.

```
def LLamado(dimension,portaa,monedas,vida):
    Busca = prueba(dimension,portaa)
    plt.connect('button_press_event', Busca.on_click)
    plt.ion()
    Busca.Pintar(portaa,monedas,vida)
    plt.draw()
    while Busca.Estado == "":
        plt.pause(0.1)
    plt.ioff()
    plt.show()
```

Graficando el tablero con la librería Matplotlib

5. Tablero de Juego

```
class Casilla:
    def __init__(self):
        self.Visible = False
        self.TieneBuque = False
        self.NumBuquesAdyacentes = 0
```

El tablero lleva la clase Casilla

```

class prueba:
    def __init__(self, tam, numBuques):
        self.Tamano = tam
        self.Tablero = []
        self.Pendientes = tam*tam
        self.Estado = ""
        self.XError = None
        self.YError = None
        for fila in range(tam):
            f = []
            for j in range(tam):
                f.append(Casilla())
            self.Tablero.append(f)
        #print(self.Tablero.append(f))
        num = 0
        while num < numBuques:
            rndx = random.randint(0,tam-1)
            rndy = random.randint(0,tam-1)
            print("Y: ", rndx+1, "X: ", rndy+1)
            #print("X: ", rndy+1)
            #print("X: ", rndx, "Y: ", rndy)
            if not self.Tablero[rndx][rndy].TieneBuque:
                self.Tablero[rndx][rndy].TieneBuque = True
                filaIni = max(rndx-1,0)
                filaFin = min(rndx+1,tam-1)
                colIni = max(rndy-1,0)
                colFin = min(rndy+1,tam-1)
                for i in range(filaIni, filaFin+1,1):
                    for j in range(colIni,colFin+1,1):
                        if i !=rndx or j != rndy:
                            self.Tablero[i][j].NumBuquesAdyacentes += 1
            num += 1

```

Para colocar los barcos de forma aleatoria

```

def Pintar(self,avv,puntos,vida):
    global lblsalida
    global lblpuntos
    global lblvidas
    #doc = open("Python/" + "Prueba" + ".txt", "w")
    if self.Estado == "G":
        #plt.suptitle("Has Ganado :D")
        MessageBox.showinfo("Felicitaciones", "Has ganado el juego ")
    elif self.Estado == "P":
        MessageBox.showinfo("Sigue intentando", "Has perdido el juego ")
        #plt.suptitle("Has Perdido :(")

```

Por último pintamos el gráfico.

```

    lblsalida = avv
    lblpuntos = puntos
    lblvidas = vida
    plt.text(0, self.Tamano + 1.5, "Buques: " + str(avv) , fontdict=None)
    plt.text(4,self.Tamano + 2, "Puntos: " + str(puntos) , fontdict=None )
    plt.text(8,self.Tamano + 2, "Vidas: " + str(vida) , fontdict=None )
    for n in range(self.Tamano+1):
        plt.plot ([0,self.Tamano],[n,n], color="black", linewidth=1)
        plt.plot ([n,n],[0,self.Tamano], color="black", linewidth=1)
    for i in range(self.Tamano):
        for j in range(self.Tamano):
            px = j + 0.5
            py = self.Tamano - (i + 0.5)
            if self.Tablero[i][j].Visible:
                if self.Tablero[i][j].TieneBuque:
                    plt.plot([px], [py], linestyle='None', marker='.', markersize=8, color='teal')
                    #doc.write([px] + '\n')
                else:
                    #if self.Tablero[i][j].NumBuquesAdyacentes != 0:
                    plt.plot([px], [py], linestyle='None', marker='.', markersize=8, color='red')
            else:
                plt.plot([px], [py], linestyle='None', marker='.', markersize=4,color='black')
    #doc.close()

```

6. Salida

```

def cerrar():
    MessageBox.showinfo("Adios", "Gracias por usar el programa")
    sys.exit()

```

Estructuras de la Fase 3

Creación del programa

1. Jugador vs Jugador

```

ventanaObtenerDimension = Toplevel()
ventanaObtenerDimension.title("Dimensiones del Tablero")
ventanaObtenerDimension.resizable(0,0)
ancho_ventana2 = 300
alto_ventana2 = 250
x_ventana2 = ventanaObtenerDimension.winfo_screenwidth() // 2 - ancho_ventana2 // 2
y_ventana2 = ventanaObtenerDimension.winfo_screenheight() // 2 - alto_ventana2 // 2
posicion2 = str(ancho_ventana2) + "x" + str(alto_ventana2) + "+" + str(x_ventana2) + "+" + str(y_ventana2)
ventanaObtenerDimension.geometry(posicion2)
labelDimension = Label (ventanaObtenerDimension, text ="Dimensiones del Tablero", font=("Verdana",16), background="#044D9A",
fg="white")
labelDimension.place(x=50, y=45)
labelNum = Label (ventanaObtenerDimension, text ="Número", font=("Verdana",16), background="#044D9A", fg="white")
labelNum.place(x=50, y=115)
textoDimension = Text(ventanaObtenerDimension, height=2, width=12 ,fg="white", font=("Consolas", 12))
textoDimension.place(x=150, y=110)
btnCrearT = Button(ventanaObtenerDimension, height=2, width=15, text="Crear", command = lambda:[GuardarConfiguracion(),Partida()])
btnCrearT.place(x=80, y=190)
ventanaObtenerDimension.withdraw()

```

Se crea una ventana en la cual se pide el nombre del jugador 2 y las dimensiones del tablero.

1.1. Tablero

```
def Partida():
    global Portaaviones, Submarino, Destructores, Buques, TotalBarcos, nuevocontador, nuevocontador2
    global totalmonedas,tokensjugador, totalmonedas2, tokensjugador2, jugador2
    global FILAS, COLUMNAS, MAR
    global BUQUE, DESTRUCTOR, SUBMARINO, DESTRUCTOR_VERTICAL, SUBMARINO_VERTICAL
    global PORTAAVIONES, PORTAAVIONES_VERTICAL, DISPARO_FALLADO, DISPARO_ACERTADO
    global JUGADOR_1, JUGADOR_2

    ventanaUser.withdraw()
    ListaAdy.crear(int(dimensionestablero))
    for x in range(int(dimensionestablero)):
        ListaAdy.insertar(x,x)
    jugador2 = nombrejugador2

    MAR = " "
    BUQUE = "B" # Ocupa una celda
    DESTRUCTOR = "D" # Ocupa dos celdas
    SUBMARINO = "S" # Ocupa tres celdas
    DESTRUCTOR_VERTICAL = "A" # Ocupa dos celdas
    SUBMARINO_VERTICAL = "C" # Ocupa tres celdas
    PORTAAVIONES = "P" # Ocupa cuatro celdas
    PORTAAVIONES_VERTICAL = "V" # Ocupa cuatro celdas
    DISPARO_FALLADO = "E"
    DISPARO_ACERTADO = "*"
    #CANTIDAD_BARCOS_INICIALES = 10
    JUGADOR_1 = usuariobusqueda
    JUGADOR_2 = jugador2
    Portaav = 1
    Subma = 2
    Destruc = 3
    Buq = 4
```

Creamos el tablero, para ello hacemos variables de los barcos, jugadores y los tipos de barcos.

```
# Formula para determinar la cantidad de barcos por tablero
# B(m) = ((m-1)/10)+1
dimens = dimensionestablero
dimen = int(dimens)
if dimen<10:
    MessageBox.showerror("Advertencia", "El Numero minimo para el tablero es de 10")
if dimen==10:
    FILAS = dimen
    COLUMNAS = dimen
    ventanaObtenerDimension.withdraw()
    textoDimension.delete(1.0, tk.END+"-1c")
    textoNombre.delete(1.0, tk.END+"-1c")
    vidas = 3
    B = int((dimen-1)/10))+1
    #print(B)
    Portaaviones = Portaav * B
    Submarino = Subma * B
    Destructores = Destruc * B
    Buques = Buq * B
    # print("Portaaviones: ", Portaaviones)
    # print("Submarinos : ", Submarino)
    # print("Destructores: ", Destructores)
    # print("Buques : ", Buques)
    TotalBarcos = Portaaviones + Submarino + Destructores + Buques
    otroB = TotalBarcos
    nuevocontador = otroB
    nuevocontador2 = otroB
    totalmonedas = monedasusuario
    pasarmoneda = totalmonedas
    tokensjugador = pasarmoneda
    totalmonedas2 = 0
    pasarmoneda2 = totalmonedas2
    tokensjugador2 = pasarmoneda2
    MessageBox.showinfo("Exito", "Tablero creado con exito")
    crearTableros()
    pruebanueva()
```

Se hacen las validaciones de las dimensiones del tablero

```
def obtener_matriz_inicial():
    matriz = []
    for y in range(FILAS):
        # Agregamos un arreglo a la matriz, que sería una fila básicamente
        matriz.append([])
        for x in range(COLUMNAS):
            # Y luego agregamos una celda a esa fila. Por defecto lleva "Mar"
            matriz[y].append(MAR)
    return matriz

# Indica si una coordenada de la matriz está vacía
def es_mar(x, y, matriz):
    return matriz[y][x] == MAR

def coordenada_en_rango(x, y):
    return x >= 0 and x <= COLUMNAS-1 and y >= 0 and y <= FILAS-1
```

Creamos la matriz para los barcos, espacios vacíos y coordenadas.

```
def colocar_e_imprimir_barcos(matriz, cantidad_barcos, jugador):
    # Dividimos y redondeamos a entero hacia abajo (ya que no podemos colocar una parte no entera de un barco)
    salidadestructores = int((Destructores)/2)
    salidasubmarinos = int((Submarino)/2)
    salidaportaav = int((Portaaviones)/2)
    destructoresverticales = Destructores - salidadestructores
    submarinosverticales = Submarino - salidasubmarinos
    portaavionesvertical = Portaaviones - salidaportaav
    barcos_una_celda = Buques
    barcos_dos_celdas_verticales = destructoresverticales
    barcos_dos_celdas_horizontales = salidadestructores
    barcos_tres_celdas_verticales = submarinosverticales
    barcos_tres_celdas_horizontales = salidasubmarinos
    barcos_cuatro_celdas_horizontiles = portaavionesvertical
    barcos_cuatro_celdas_verticales = salidaportaav

    if jugador == JUGADOR_1:
        print("")
    else:
        print("")

    # Primero colocamos los de cuatro celdas para que se acomoden bien
    if barcos_cuatro_celdas_verticales == 0:
        matriz = colocar_barcos_de_cuatro_celdas_horizontal(
            barcos_cuatro_celdas_horizontiles, PORTAAVIONES, matriz)
        #matriz = colocar_barcos_de_cuatro_celdas_vertical(
        #    barcos_cuatro_celdas_verticales, PORTAAVIONES_VERTICAL, matriz)
        matriz = colocar_barcos_de_tres_celdas_horizontal(
            barcos_tres_celdas_horizontales, SUBMARINO, matriz)
        matriz = colocar_barcos_de_tres_celdas_vertical(
            barcos_tres_celdas_verticales, SUBMARINO_VERTICAL, matriz)
        matriz = colocar_barcos_de_dos_celdas_horizontal(
            barcos_dos_celdas_horizontales, DESTRUCTOR, matriz)
        matriz = colocar_barcos_de_dos_celdas_vertical(
            barcos_dos_celdas_verticales, DESTRUCTOR_VERTICAL, matriz)
        matriz = colocar_barcos_de_una_celda(barcos_una_celda, BUQUE, matriz)
```

Colocamos los barcos de forma que no se ponga uno encima de otro, distribuyendo entre verticales y horizontales.

```

def colocar_barcos_de_una_celda(cantidad, tipo_barco, matriz):
    barcos_colocados = 0
    while True:
        x = obtener_x_aleatoria()
        y = obtener_y_aleatoria()
        if es_mar(x, y, matriz):
            matriz[y][x] = tipo_barco
            barcos_colocados += 1
        if barcos_colocados >= cantidad:
            break
    return matriz

def colocar_barcos_de_dos_celdas_horizontal(cantidad, tipo_barco, matriz):
    barcos_colocados = 0
    while True:
        x = obtener_x_aleatoria()
        y = obtener_y_aleatoria()
        x2 = x+1
        if coordenada_en_rango(x, y) and coordenada_en_rango(x2, y) and es_mar(x, y, matriz) and es_mar(x2, y, matriz):
            matriz[y][x] = tipo_barco
            matriz[y][x2] = tipo_barco
            barcos_colocados += 1
        if barcos_colocados >= cantidad:
            break
    return matriz

```

Poniendo los barcos en celdas aleatorias. Esto sucesivamente con todos los tipos de barcos.

```

def imprimir_matriz(matriz, deberia_mostrar_barcos, jugador):
    doc = open("Jugador1.txt", "w")
    letra = 1
    for y in range(FILAS):
        for x in range(COLUMNAS):
            celda = matriz[y][x]
            valor_real = celda
            if not deberia_mostrar_barcos and valor_real != MAR and valor_real != DISPARO_FALLADO and valor_real != DISPARO_ACERTADO:
                if valor_real == BUQUE:
                    valor_real = "B"
                elif valor_real == SUBMARINO:
                    valor_real = "S"
                elif valor_real == DESTRUCTOR:
                    valor_real = "D"
                elif valor_real == DESTRUCTOR_VERTICAL:
                    valor_real = "A"
                elif valor_real == SUBMARINO_VERTICAL:
                    valor_real = "C"
                elif valor_real == PORTAAVIONES:
                    valor_real = "P"
                elif valor_real == PORTAAVIONES_VERTICAL:
                    valor_real = "V"
                else:
                    valor_real = "X"
            doc.write(f'{valor_real}')
            letra+=1
        doc.write("\n")
    doc.close()
    with open('Jugador1.txt') as archivo:
        l = 0
        c = 0
        lineas = archivo.readlines()
        for linea in lineas:
            columnas = linea
            l += 1
            for col in columnas:
                if col != '\n':

```

Creamos un txt donde llevamos el control de la matriz y este es ejecutado para mostrar la matriz en la ventana. Esto para cada jugador.

```

with open('Jugador2.txt') as archivo1:
    l1 = 0
    c1 = 0
    lineas1 = archivo1.readlines()
    for linea1 in lineas1:
        columnas1 = linea1
        l1 += 1
        for col1 in columnas1:
            if col1 != '\n':
                c1 += 1
                matriz2.insert(l1, c1, col1)
        c1 = 0
    matriz2.graficarNeato2('Jugador2')

```

```

def solicitando_coordenadas():
    global XDisparo, YDisparo

    y = None
    x = None

    while True:
        try:
            y = ataqueY
            y2 = ataqueY
            if coordenada_en_rango(y-1, 0):
                y = y-1
                YDisparo = y

            break
        else:
            print("Fila invalida")
        except:
            print("Ingrese una coordenada valida")
    while True:
        try:
            x = ataqueX
            x2 = ataqueX
            if coordenada_en_rango(x-1, 0):
                x = x-1
                XDisparo = x
            break
        else:
            print("Columna invalida")
        except:
            print("Ingrese una coordenada valida")
    ListaAdy.conexion(XDisparo, YDisparo)
    disparar()

```

Creando un metodo para las coordenadas donde se va a ejecutar un disparo.

```

def disparar() -> bool:
    global nuevocontador, tokensjugador, barcosdestruidos, tokenganados, disparosfalladosj1
    disparosfalladosj1 = 0
    barcosdestruidos = 0
    tokenganados = 0
    x = int(YDisparo)
    y = int(XDisparo)
    matriz = MatrizActual
    if es_mar(x, y, matriz):
        matriz[y][x] = DISPARO_FALLADO
        #print("Fallado")
        MessageBox.showinfo("Fallo", "Disparo fallado")
        disparosfalladosj1 += 1
        imprimir_matriz(MatrizActual, False,
                         oponente_de_jugador(turno_actual1))
        return False
    # Si ya habia disparado antes, se le cuenta como falla igualmente
    elif matriz[y][x] == DISPARO_FALLADO or matriz[y][x] == DISPARO_ACERTADO:
        #print("De nuevo")
        #MessageBox.showinfo("Fallo", "El disparo ya ha sido fallado")
        disparosfalladosj1 += 1
        imprimir_matriz(MatrizActual, False,
                         oponente_de_jugador(turno_actual1))
        return False
    else:
        matriz[y][x] = DISPARO_ACERTADO
        MessageBox.showinfo("Exito", "El disparo ha impactado en un barco")
        nuevocontador -= 1
        tokensjugador += 20
        barcosdestruidos += 1
        tokenganados += 20
        #print("Acertado")
        imprimir_matriz(MatrizActual, False,
                         oponente_de_jugador(turno_actual1))
        return True

```

Usamos una función booleana para retornar un “True” o “False” en caso se haya fallado o no el disparo.

```

def abrirventana():
    global ventanaNueva, lbltokens
    lblbarcoos = nuevocontador
    lbltokens = tokensjugador
    ventanaNueva = Toplevel()
    ventanaNueva.title("Tablero Jugador 1")
    ventanaNueva.resizable(0,0)
    ancho_ventanaN = 1100
    alto_ventanaN = 800
    x_ventanaN = ventanaNueva.winfo_screenwidth() // 2 - ancho_ventanaN // 2
    y_ventanaN = ventanaNueva.winfo_screenheight() // 2 - alto_ventanaN // 2
    posicionN = str(ancho_ventanaN) + "x" + str(alto_ventanaN) + "+" + str(x_ventanaN) + "+" + str(y_ventanaN)
    ventanaNueva.geometry(posicionN)
    image = Image.open("Salida/matriz_Jugador1.png")
    resize_image = image.resize((700, 700))
    img = ImageTk.PhotoImage(resize_image)
    labelPhotoN = Label(ventanaNueva, image=img)
    labelPhotoN.image = img
    labelPhotoN.place(x=250,y=30)
    #labelPhotoN.pack()
    labelusuarioactual = Label (ventanaNueva, text ="Turno de : " + JUGADOR_1, font=("Verdana",16), background="#044D9A", fg="white")
    labelusuarioactual.place(x=550, y=0)
    labelAtacando = Label (ventanaNueva, text ="Atacando a : " + JUGADOR_2, font=("Verdana",16), background="#044D9A", fg="white")
    labelAtacando.place(x=20, y=100)
    labelPuntos = Label (ventanaNueva, text ="Tokens: " + str(lbltokens), font=("Verdana",16), background="#044D9A", fg="white")
    labelPuntos.place(x=20, y=130)
    labelBarcos = Label (ventanaNueva, text ="Barcos Restantes: " + str(lblbarcoos), font=("Verdana",16), background="#044D9A", fg="white")
    labelBarcos.place(x=20, y=160)
    labelBuques = Label (ventanaNueva, text ="Buques: " + str(Buques), font=("Verdana",16), background="#044D9A", fg="white")
    labelBuques.place(x=20, y=190)
    labelDestructor = Label (ventanaNueva, text ="Destructores: " + str(Destructores), font=("Verdana",16), background="#044D9A", fg="white")
    labelDestructor.place(x=20, y=220)
    labelSubmarinos = Label (ventanaNueva, text ="Submarinos: " + str(Submarino), font=("Verdana",16), background="#044D9A", fg="white")
    labelSubmarinos.place(x=20, y=250)
    labelPortaa = Label (ventanaNueva, text ="Portaaviones: " + str(Portaaviones), font=("Verdana",16), background="#044D9A", fg="white")
    labelPortaa.place(x=20, y=280)
    labelPosX = Label (ventanaNueva, text ="Pos X", font=("Verdana",16), background="#044D9A", fg="white")

```

Creamos una ventana en donde se visualiza el tablero, contadores de barcos destruidos, el jugador que está atacando, entre otros.

```

def pruebanueva():
    global MatrizActual, turno_actual1, jugadorganador1, tokenganador1, barcosganador1, falladosj1
    DERROTADO = FALSE
    turno_actual1 = JUGADOR_1
    while DERROTADO == False:
        if turno_actual1:
            matriz_oponente = matriz_j1
            imprimir_matriz(matriz_oponente, False,
                            oponente_de_jugador(turno_actual1))
            MatrizActual = matriz_oponente
            abrirventana()
            if nuevocontador == 0 :
                jugadorganador1 = turno_actual1
                tokenganador1 = tokenganados
                barcosganador1 = barcosdestruidos
                falladosj1 = disparosfalladosj1
                MessageBox.showinfo("Exito", "El jugador " + jugadorganador1 + " ha ganado")
                agregarpuntos()
                resultadojugador1()
                tableroefinal1()
                ventanaNueva.destroy()
                mostrarjugador1()
                break
            DERROTADO = TRUE

```

Creando un ciclo que cuando el total de barcos sea distinto de 0 no termine, de lo contrario entra al ciclo como el ganador.

```
def mostrarjugador1():
    ventanaGanador = Toplevel()
    ventanaGanador.title("Jugador Logeado")
    ventanaGanador.resizable(0,0)
    ancho_ventanaGanador = 400
    alto_ventanaGanador = 400
    x_ventanaGanador = ventanaGanador.winfo_screenwidth() // 2 - ancho_ventanaGanador // 2
    y_ventanaGanador = ventanaGanador.winfo_screenheight() // 2 - alto_ventanaGanador // 2
    posicionNombre = str(ancho_ventanaGanador) + "x" + str(alto_ventanaGanador) + "+" + str(x_ventanaGanador) + "+" + str(y_ventanaGanador)
    ventanaGanador.geometry(posicionNombre)
    labelNombreGanador= Label (ventanaGanador, text ="Resultados del usuario : " + usuariobusqueda, font=("Verdana",16), background="#044D9A", foreground="white")
    labelNombreGanador.place(x=75, y=30)
    labelDestruidos = Label (ventanaGanador, text ="Barcos Destruidos : " + str(barcosdestruidos), font=("Verdana",16), background="#044D9A", foreground="white")
    labelDestruidos.place(x=50, y=90)
    labelTokensG = Label (ventanaGanador, text ="Tokens Ganados : " + str(tokenganados), font=("Verdana",16), background="#044D9A", foreground="white")
    labelTokensG.place(x=50, y=120)
    btnDisparos = Button(ventanaGanador, height=2, width=12, text="Ver tablero", command = lambda:[abrirtablerefinal1()], background="#044D9A", foreground="white")
    btnDisparos.place(x=50, y=200)
    btnListaA = Button(ventanaGanador, height=2, width=12, text="Lista de Jugadas", command = lambda:[abrirlistaadyacencia1()], background="#044D9A", foreground="white")
    btnListaA.place(x=200, y=200)
    btnGrafo = Button(ventanaGanador, height=2, width=12, text="Grafo de Jugadas", command = lambda:[abrirgrafoLista()], background="#044D9A", foreground="white")
    btnGrafo.place(x=50, y=250)
    btnVolverJuego = Button(ventanaGanador, height=2, width=12, text="Volver a jugar", command = lambda:[Partida()], background="#044D9A", foreground="white")
    btnVolverJuego.place(x=50, y=300)
    btnEstadisticas = Button(ventanaGanador, height=2, width=16, text="Estadisticas del Ganador", command = lambda:[estadisticasganadoras()])
    btnEstadisticas.place(x=200, y=250)
    btnRegresarM = Button(ventanaGanador, height=2, width=12, text="Regresar al menu", command = lambda:[ventanaGanador.withdraw()], background="#044D9A", foreground="white")
    btnRegresarM.place(x=200, y=300)
```

Creando una nueva ventana que se abre al finalizar la partida.

```
def resultadojugador1():
    ListaAdy.GraficoLista()
    ListaAdy.GrafoLista()
```

Obteniendo la lista de adyacencia.

1.2. Lista de Adyacencia

```
class ENodo():
    def __init__(self, index):
        self.index = index
        self.siguiente = None

class VNodo():
    def __init__(self):
        self.dato = None
        self.inicio = None

    def insertar(self, index):
        nuevo = ENodo(index)
        if self.inicio == None:
            self.inicio = nuevo
        else:
            aux = self.inicio
            while True:
                if aux.siguiente == None:
                    aux.siguiente = nuevo
                    break
                aux = aux.siguiente
```

Creando los nodos de la lista.

```
class ListaDG():
    def __init__(self):
        self.v = []

    def crear(self, tamano):
        for i in range(tamano):
            self.v.append(i)
            self.v[i] = VNodo()

    def insertar(self,dato,pos):
        if pos >= 0 and pos < len(self.v):
            self.v[pos].dato = dato
            self.v[pos].inicio = None

    def conexion(self,inicio,fin):
        if inicio >= 0 and inicio < len(self.v):
            self.v[inicio].insertar(fin)
```

Creando el tamaño de la lista, las posiciones y sus respectivas conexiones.

```
def GraficoLista(self):
    contenido = ''' digraph G {\n    graph [rankdir = LR ]\n    node [ style=filled,shape = box, fillcolor="white"]\n    ...\n    rank = '{rank = same; '\n    label = ''\n    apuntarsiguiente = ''\n    for sig in self.v:\n        rank += "\\" + "nod" + str(sig.dato +1) + "nod\\ " \n        label += "\\" + "nod" + str(sig.dato +1) + "nod\\ " + "[label = \\" + str(sig.dato +1) + "\\"]\\n"\n        apuntarsiguiente += "\\" + "nod" + str(sig.dato +1) + "nod\\ " + "->"\n    contenido += rank + '};\\n'\n    contenido += label + '\\n'\n    apuntarsiguiente = apuntarsiguiente[:-2]\n    contenido += apuntarsiguiente + '\\n'\n    for sig in self.v:\n        aux = sig.inicio\n        apuntarsiguiente = ''\n        count = 0\n        while aux != None:\n            contenido += 'nod' + str(sig.dato +1) + 'nod' + str(self.v[aux.index].dato + 1) + ' [label = \\" + str(self.v[aux.index].dato + 1) + "\\"]\n            apuntarsiguiente += 'nod' + str(sig.dato +1) + 'nod' + str(self.v[aux.index].dato + 1) + '->\n            aux = aux.siguiente\n            count += 1\n\n        if count > 0:\n            apuntarsiguiente = apuntarsiguiente[:-2]\n            contenido += 'nod' + str(sig.dato +1) + 'nod->' + apuntarsiguiente + '\\n'\n    contenido += '}\n\n    dot = "Graficos/ListaAdy1.dot"\n    with open(dot, 'w') as f:\n        f.write(contenido)\n    result = "Salida/ListaAdy1.png"\n    os.system('dot -Tpng ' + dot + ' -o' + result)'''
```

De este mismo se obtiene su salida con graphviz

1.3. Grafo de la lista

```
def GrafoLista(self):
    contenido = ''' digraph G {\n'''
    for sig in self.v:
        auxiliar = sig.inicio
        while auxiliar != None:
            contenido += str(sig.dato+1) + '->' + str(self.v[auxiliar.index].dato + 1) + ';\n'
            auxiliar = auxiliar.siguiente
    contenido += '}'\n\n
    dot = "Graficos/GrafoLista.dot"
    with open(dot, 'w') as f:
        f.write(contenido)
    result = "Salida/GrafoLista.png"
    os.system('dot -Tpng ' + dot + ' -o' + result)
```

Obtenemos la salida del grafo de la lista anterior.