

Universidad de San Carlos de Guatemala  
Facultad de ingeniería  
Ingeniería en Ciencias y Sistemas  
ESTRUCTURA DE DATOS



## MANUAL TECNICO

Alberto Josue Hernández Armas 201903553

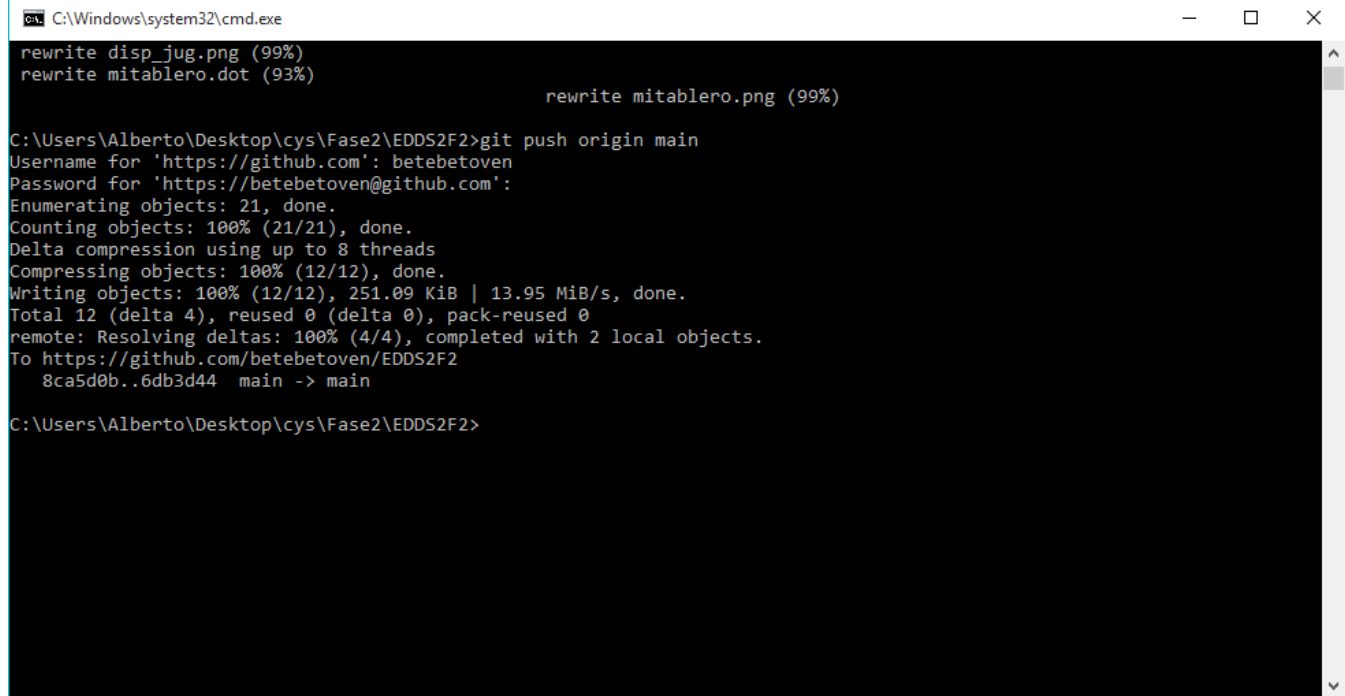
Guatemala 3 de OCTUBRE de  
2022

## **INTRODUCCION**

En el presente manual técnico podremos observar cómo es que está diseñado el código y explicando que forman la funcionalidad de una manera general, teniendo como objetivo principal que sea más entendible. Determinando los métodos utilizados y explicando algunas palabras claves refiriéndonos a sus propiedades demostrando cuál es su función dentro de los bloques de código que se nos presentan a continuación.

## MANUAL TECNICO

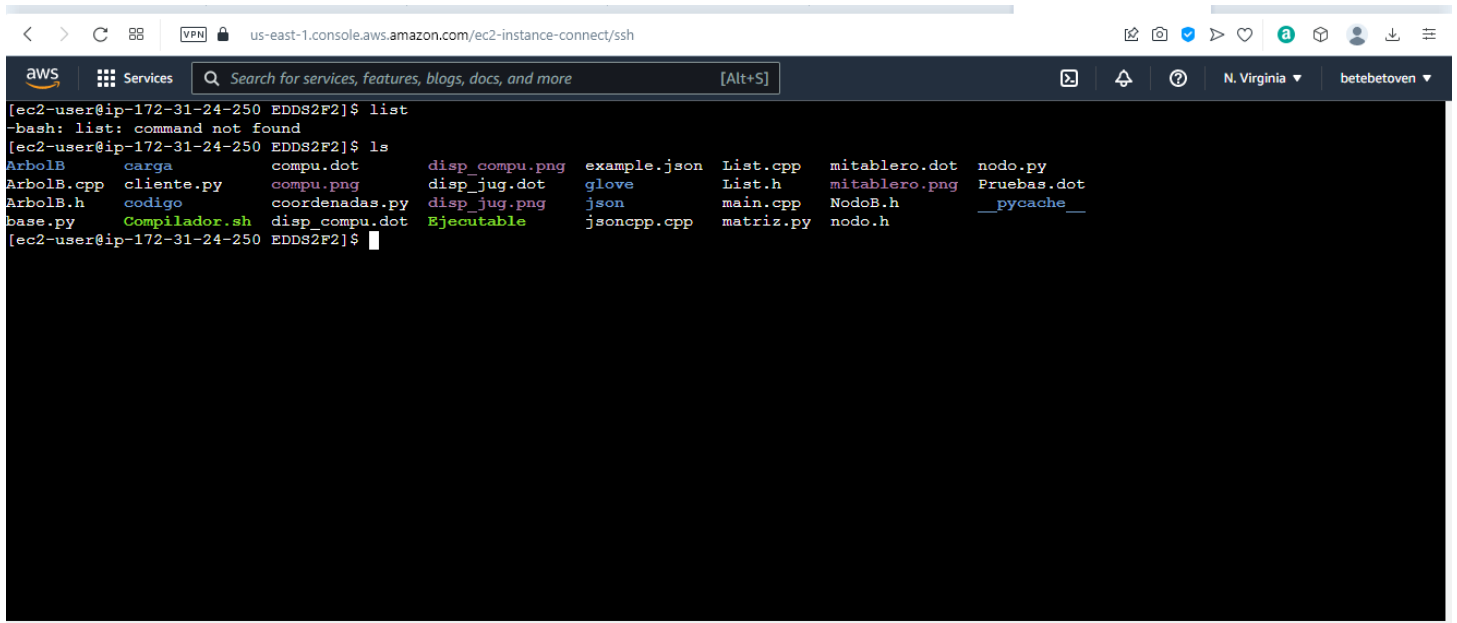
El proyecto se trabajo como una API, de manera que se tiene un servidor que se encarga del manejo de la información, y un cliente que se encarga d ingresarla, recibirla y hasta cierto punto procesarla. En este caso debido a que la REST API era en C++, hacerlo en Windows representaba demasiados problemas y particionar el disco tomaría demasiado tiempo, por lo que se utilizó el servicio de aws, para tener una máquina virtual que sosteniera el servidor en c++,y conectándonos a ella por medio de su IP, para hacer las llamadas desde el cliente.

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Windows\system32\cmd.exe'. The window contains the following text: 'rewrite disp\_jug.png (99%)', 'rewrite mitablero.dot (93%)', 'rewrite mitablero.png (99%)', 'C:\Users\Alberto\Desktop\cys\Fase2\EDDS2F2>git push origin main', 'Username for 'https://github.com': betebetoven', 'Password for 'https://betebetoven@github.com':', 'Enumerating objects: 21, done.', 'Counting objects: 100% (21/21), done.', 'Delta compression using up to 8 threads', 'Compressing objects: 100% (12/12), done.', 'Writing objects: 100% (12/12), 251.09 KiB | 13.95 MiB/s, done.', 'Total 12 (delta 4), reused 0 (delta 0), pack-reused 0', 'remote: Resolving deltas: 100% (4/4), completed with 2 local objects.', 'To https://github.com/betetbetoven/EDDS2F2', '8ca5d0b..6db3d44 main -> main', and 'C:\Users\Alberto\Desktop\cys\Fase2\EDDS2F2>'.

```
C:\Windows\system32\cmd.exe
rewrite disp_jug.png (99%)
rewrite mitablero.dot (93%)
rewrite mitablero.png (99%)

C:\Users\Alberto\Desktop\cys\Fase2\EDDS2F2>git push origin main
Username for 'https://github.com': betebetoven
Password for 'https://betebetoven@github.com':
Enumerating objects: 21, done.
Counting objects: 100% (21/21), done.
Delta compression using up to 8 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (12/12), 251.09 KiB | 13.95 MiB/s, done.
Total 12 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), completed with 2 local objects.
To https://github.com/betetbetoven/EDDS2F2
8ca5d0b..6db3d44 main -> main

C:\Users\Alberto\Desktop\cys\Fase2\EDDS2F2>
```

A screenshot of an AWS console terminal window. The browser address bar shows 'us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh'. The terminal interface has an AWS logo and a search bar. The terminal text shows a user at 'ec2-user@ip-172-31-24-250 EDD82F2' running 'list' (which fails) and 'ls' (which lists files). The files listed are: ArbolB, ArbolB.cpp, ArbolB.h, base.py, carga, cliente.py, codigo, Compilador.sh, compu.dot, compu.png, coordenadas.py, disp\_compu.dot, disp\_compu.png, disp\_jug.dot, disp\_jug.png, Ejecutable, example.json, glove, json, jsoncpp.cpp, List.cpp, List.h, main.cpp, matriz.py, mitablero.dot, mitablero.png, Nodob.h, nodo.py, Pruebas.dot, and \_\_pycache\_\_.

```
[ec2-user@ip-172-31-24-250 EDD82F2]$ list
-bash: list: command not found
[ec2-user@ip-172-31-24-250 EDD82F2]$ ls
ArbolB      carga      compu.dot  disp_compu.png  example.json  List.cpp    mitablero.dot  nodo.py
ArbolB.cpp  cliente.py compu.png   disp_jug.dot    glove         List.h      mitablero.png  Pruebas.dot
ArbolB.h    codigo     coordenadas.py disp_jug.png    json          main.cpp    Nodob.h        __pycache__
base.py     Compilador.sh disp_compu.dot Ejecutable      jsoncpp.cpp   matriz.py   nodo.h
```

Luego de configurar la maquina virtual, se procedio a utilizar la maquina principal como un editor de texto, subiendo el código a github, y haciendo un pull desde la maquina virtual, evitándonos contratar la interfaz grafica de aws y facilitando la modificación de datos. En si aws quedo como un receptor de todos los cambios que se realizaban en github.

Las estructura principales del proyecto se mantienen iguales, ya que el servidor se basa en la fase 1 del proyecto, las estructuras nuevas son El árbol B y la matriz dispersa, El árbol be se implementa de la siguiente manera:

```

3  //
4
5  #ifndef F1_ARBOLB_H
6  #define F1_ARBOLB_H
7  #include <algorithm>
8
9  #include "NodoB.h"
10
11 template<typename T>
12 class ArbolB {
13 public:
14     int orden_arbol = 5;
15     NodoB<T> *raiz;
16
17     ArbolB() {
18         raiz = NULL;
19     }
20     void insertar(T id);
21     pair<NodoB<T>*, pair<bool, bool>> insertarCrearRama(NodoB<T>* nodo, NodoB<T>* rama);
22     NodoB<T>* dividir(NodoB<T>* rama);
23     pair<NodoB<T>*, bool> insertarEnRama(NodoB<T>* primero, NodoB<T>* nuevo);
24     bool esHoja(NodoB<T>* primero);
25     int contador(NodoB<T>* primero);
26     string Grafo();
27     string GrafoArbolAbb(NodoB<T>*rama);
28     string GrafoRamas(NodoB<T>*rama);
29     string GrafoConexionRamas(NodoB<T>*rama);
30 private:
31
32 };
33
34 #endif //F1_ARBOLB_H

```

Siendo estos los recursos que se utilizaran. Este archivo es su header.

```

1  #include "ArbolB.h"
2  #include "json/json.h"
3  // #include <windows.h>
4  #include <fstream>
5  #include <string>
6  #include <sstream>
7  #include <iostream>
8  template<typename T>
9  void ArbolB<T>::insertar(T id) {
10     NodoB<T>* nodo = new NodoB<T>(id);
11     if (raiz == NULL) {
12         raiz = nodo;
13     } else {
14         pair< NodoB<T>*, pair<bool, bool>> ret = insertarCrearRama(nodo, raiz);
15         NodoB<T>* obj = ret.first;
16         if ((ret.second.first or ret.second.second) and obj != NULL) { //si se divide la rama o se inserta al inicio, la raiz cambia
17             cout << "se cambia de rama principal ID:" << obj->id << "\n";
18             raiz = obj;
19         }
20     }
21 }
22

```

Aca veremos su implementación, en su archivo cpp, y el algoritmo de ordenamiento de información por cada rama.

```

template<typename T>
pair<NodoB<T>*, pair<bool, bool>> ArbolB<T>::insertarCrearRama(NodoB<T>* nodo, NodoB<T>* rama) {
    pair< NodoB<T>*, pair<bool, bool>> ResultadoRama;
    ResultadoRama.first = NULL; //nodo Inicial de la rama
    ResultadoRama.second.first = false; //indica si se dividió la rama
    ResultadoRama.second.second = false; //indica si se modifica el inicio de la rama
    if (esHoja(rama)) { //si el nodo es hoja se inserta directamente dentro de ella
        pair< NodoB<T>*, bool> resultado = insertarEnRama(rama, nodo); //insertamos el nuevo elemento dentro de la rama actual
        ResultadoRama.first = resultado.first; //posee la rama con el valor ya insertado
        ResultadoRama.second.second = resultado.second; //posee el resultado de si se modifico el inicio en el insert anterior
        if (contador(resultado.first) == orden_arbol) { //si la rama posee mas elementos de los permitidos se divide
            cout << "La rama debe dividirse\n";
            ResultadoRama.first = dividir(resultado.first); //dividimos la rama y obtenemos una nueva rama con sus respectivos apuntadores
            ResultadoRama.second.first = true; //identificar que la rama se dividió
        }
    } else { //si el nodo es rama se debe buscar la posicion donde insertarlo
        NodoB<T>*temp = rama;
        do {
            if (nodo->value["nick"].asString() == temp->value["nick"].asString()) { //valor ya insertado, no se permiten repeditos
                cout << "insertarCrearRama(), El ID " << nodo->value["nick"].asString() << " ya existe\n"; //donde dice value.nombre cabiar a las instancias del JSON:value
                return ResultadoRama;
            } else if (nodo->id < temp->id) {
                pair< NodoB<T>*, pair<bool, bool>> ResultadoInsert = insertarCrearRama(nodo, temp->izquierda);
                if (ResultadoInsert.second.second and ResultadoInsert.first != NULL) { //si se modifico el inicio de la rama
                    ResultadoRama.second.second = true;
                    temp->izquierda = ResultadoInsert.first;
                }
                if (ResultadoInsert.second.first) { //se dividió la subrama
                    pair< NodoB<T>*, bool> auxInsert = insertarEnRama(rama, ResultadoInsert.first);
                    rama = auxInsert.first;
                    if (auxInsert.second) {
                        ResultadoRama.first = rama;
                    }
                }
            }
        } while (temp != NULL);
    }
}

```

```

70         if (contador(rama) == orden_arbol) {
71             ResultadoRama.first = dividir(rama);
72             ResultadoRama.second.first = true;
73         }
74     }
75     return ResultadoRama;
76 } else if (temp->siguiente == NULL) {
77     pair < NodoB<T>*, pair<bool, bool>> ResultadoInsert = insertarCrearRama(nodo, temp->derecha);
78     if (ResultadoInsert.second.second and ResultadoInsert.first != NULL) { //si se modifico el inicio de la rama
79         ResultadoRama.second.second = true;
80         temp->derecha = ResultadoInsert.first;
81     }
82     if (ResultadoInsert.second.first) { //se dividió la subrama
83         pair < NodoB<T>*, bool> auxInsert = insertarEnRama(rama, ResultadoInsert.first);
84         rama = auxInsert.first;
85         if (auxInsert.second) {
86             ResultadoRama.first = rama;
87         }
88         if (contador(rama) == orden_arbol) {
89             ResultadoRama.first = dividir(rama);
90             ResultadoRama.second.first = true;
91         }
92     }
93     return ResultadoRama;
94 }
95     temp = temp->siguiente;
96 } while (temp != NULL);
97 }
98 return ResultadoRama;
99 }
100

```

Implementacion de la división de una rama:

```

template<typename T>
NodoB<T>* ArbolB<T>::dividir(NodoB<T>* rama) {
    //int val = -999;
    //T val = T("nulo");
    T val;
    //val.id = -999;
    NodoB<T>*temp = NULL;
    NodoB<T>*Nuevito = NULL;
    NodoB<T>*aux = rama;

    NodoB<T>*rderecha = NULL;
    NodoB<T>*rizquierda = NULL;

    int cont = 0;
    while (aux != NULL) {
        cont++;
        //implementacion para dividir unicamente ramas de 4 nodos
        if (cont < 3) {
            //val = aux->id;
            val = aux->value;
            temp = new NodoB<T>(val);
            temp->izquierda = aux->izquierda;
            if (cont == 2) {
                temp->derecha = aux->siguiente->izquierda;
            } else {
                temp->derecha = aux->derecha;
            }
            rizquierda = insertarEnRama(rizquierda, temp).first;
        } else if (cont == 3) {

```

Método de inserción de rama:



```

pair<NodoB<T>*, bool> ArbolB<T>::insertarEnRama(NodoB<T>* primero, NodoB<T>
    pair < NodoB<T>*, bool> ret;
    ret.second = false;
    if (primero == NULL) {
        //primero en la lista
        ret.second = true;
        primero = nuevo;
    } else {
        //recorrer e insertar
        NodoB<T>* aux = primero;
        while (aux != NULL) {
            if (aux->value["nick"].asString() == nuevo->value["nick"].asStr
                cout << "insertarEnRama(), El ID " << nuevo->id << " ya exi
                break;
            } else {
                if (aux->id > nuevo->id) {
                    if (aux == primero) {//----->insertar al inicio
                        aux->anterior = nuevo;
                        nuevo->siguiente = aux;
                        //ramas del nodo
                        aux->izquierda = nuevo->derecha;
                        nuevo->derecha = NULL;
                        ret.second = true;
                        primero = nuevo;
                        break;
                    } else {//----->insertar en medio;
                        nuevo->siguiente = aux;
                        //ramas del nodo
                        aux->izquierda = nuevo->derecha;
                        nuevo->derecha = NULL;

                        nuevo->anterior = aux->anterior;
                        aux->anterior->siguiente = nuevo;
                        aux->anterior = nuevo;
                    }
                }
            }
        }
    }
}

```

Por medio de estos métodos de inserción y de división se logra el ordenamiento adecuando

A continuación se presenta la implementación de la matriz dispersa, esta se realizó en el lenguaje python. Primero tenemos el nodo que guardará toda la información que el brindemos, así como en c++ se utilizaron las estructuras en modo template para poder reutilizarlas en el contexto que nosotros queramos, python nos permite trabajar las variables con un tipo de dato dinámico, por lo que no permite seguir en la misma línea de pensamiento.

```
15 lines (13 sloc) | 331 Bytes

1  from coordenadas import coordenadas
2
3
4  class nodo:
5      arriba= None
6      abajo = None
7      derecha = None
8      izquierda = None
9      c = None
10     barco = ""
11     def __init__(self, barco,x,y):
12         self.barco = barco
13         self.c = coordenadas(x,y)
14     def __str__(self):
15         return '{c:[' + str(self.c) + '],b:' + self.barco+'}'
```

La ventaja de python es que puede darle una representación string directa a los objetos que creamos, a diferencia de c++.

Implementación de la matriz:

```
1  from tkinter.messagebox import NO
2  from nodo import nodo
3  import random
4  import pyperclip
5  import os
6
7  class par:
8      x = 0
9      y=0
10     def __init__(self,x,y):
11         self.x = x
12         self.y = y
13     def __str__(self):
14         return '[x:' + str(self.x) + ',y:' + str(self.y)+']'
15
16 class matriz:
17     raiz = nodo("root", -1,-1)
18     dx = 0
19     dy = 0
20     ocupados = []
21     general = "digraph G\n"+"{label=\"expresion regular\"\n"+          node[shape = circle]\n"+          node[style = filled]\n"+          node[fillcolor = \"#EEEEEE\"]\n"
22     espacios = {
23         "pt":4,
24         "sub":3,
25         "dt":2,
26         "b":1
27     }
```

La matriz es una organizacion en la cual se trabajan con nodos ejes, que ayudan a localizar la información por su coordenada, sin instanciar los espacios vacios para evitar un gasto innecesario de la memoria

```
def __init__(self,t ):
    self.dx = t
    self.dy = t
    self.raiz = nodo("root", -1,-1)
    self.ocupados = []
    self.creatodo()

def recursivx(self,rooot, cont, meta):
    if cont == meta:
        print(str(rooot))
        return
    else:
        rooot.derecha = nodo("ejex",cont,-1)
        rooot.derecha.izquierda = rooot
        cont = cont+1
        print(str(rooot))
        self.recursivx(rooot.derecha,cont,meta)

def recursivy(self,rooot, cont, meta):
    if cont == meta:
        print(str(rooot))
        return
    else:
        rooot.abajo = nodo("ejey",-1,cont)
        rooot.abajo.arriba = rooot
        cont = cont +1
        print(str(rooot))
        self.recursivy(rooot.abajo,cont,meta )

def creatodo(self):
    self.recursivx(self.raiz,0,self.dx)
    self.recursivy(self.raiz,0,self.dy)
```

El método de icializacion de encarga de crear los ejes para que la información sea ingresaa de una manera más fácil , el tamaño de los ejes puede ser dinamico si se lo desea, pero en este caso no es asi.

El ingreso a la matriz es por medio de coordenadas, además de que ingresa automáticamente los barcos.

```
65     def ingresar(self,x,y,barco):
66         for n in self.ocupados:
67             if(n.x == x and n.y == y):
68                 return False
69         if(x>=self.dx or y >= self.dy or x<0 or y < 0):
70             return False
71         nuevo_nodo = nodo(barco,x,y)
72         print("ingresando: "+str(nuevo_nodo))
73         ahora = self.raiz
74         while(ahora.c.x != x):
75
76             ahora = ahora.derecha
77
78         while(ahora != None):
79
80             if(ahora.abajo == None and ahora.c.y < y):
81                 ahora.abajo = nuevo_nodo
82                 ahora.abajo.arriba = ahora
83
84             elif(ahora.abajo!= None and ahora.abajo.c.y >y and ahora.c.y < y):
85                 aux = ahora.abajo
86                 ahora.abajo = nuevo_nodo
87                 ahora.abajo.arriba = ahora
88                 ahora.abajo.abajo = aux
89                 ahora.abajo.abajo.arriba = ahora.abajo
90             ahora = ahora.abajo
91         ahora = self.raiz
92         #ahora toca de lado de y para ingresar en x
93         while(ahora.c.y != y):
94
95             ahora = ahora.abajo
96
97         while(ahora != None):
98             if(ahora.derecha == None and ahora.c.x < x):
99                 ahora.derecha = nuevo_nodo
```

También se pueden eliminar nodos sin afectar su entorno:

```

111         ahora = ahora.derecha
112     def eliminar(self,x,y):
113         if(x>self.dx or y> self.dy):
114             return False          #SOLO SE UTILIZAN DOS CASOS EN LA ELIMINACION YA QUE DE FIJO
115         bandera = False          #SIEMPRE VA A TENER NODO A LA IZQUIERDA Y ARRIBA, POR LO QUE PUEDE FACTORIZARSE
116         for n in self.ocupados:#A UN CASO GENERAL
117             if(n.x == x and n.y == y):
118                 bandera = True
119         if bandera == True:
120             ahora = self.raiz
121             while(ahora.c.x != x):
122                 ahora = ahora.derecha
123             while(ahora.c.y != y):
124                 ahora = ahora.abajo
125             ahora.arriba.abajo = ahora.abajo
126             if(ahora.abajo != None):
127                 ahora.abajo.arriba = ahora.arriba
128             ahora.izquierda.derecha = ahora.derecha
129             if(ahora.derecha != None):
130                 ahora.derecha.izquierda = ahora.izquierda
131             ahora= None
132             for n in self.ocupados:
133                 if(n.x == x and n.y == y):
134                     self.ocupados.remove(n)
135         return bandera
136

```

El resto de métodos es para la realización del videojuego.

Luego tenemos la creación, instancia e implementación de la interfaz gráfica, que es donde se llevara a cabo todas las implementaciones y conexiones con el servidor.

```

15 from matriz import matriz
16
17
18 from PIL import ImageTk, Image
19
20 tablero_jugador_global = None
21 tablero_computadora_global = None
22 tablero_disparos_computadora_global = None
23 tablero_disparos_jugador_global = None
24 direccion = "one"
25 base_url = "http://3.88.228.81:8080/"
26 def entrada():
27     global direccion
28     direccion = askopenfilename()
29     f = open(direccion, "r")
30     return f.read()
31 def carga_masiva(entrada):
32     res = requests.post(f'{base_url}/Lista/{entrada}')
33     data = res.text#convertimos la respuesta en dict
34
35     f = open(f'arbolb.dot', "w")
36     f.write(data)
37     f.close()
38     os.system(f'dot -Tpng arbolb.dot -o arbolb.png')
39
40
41     ver5()
42     print(data)
43 def login(usuario, contraseña):
44     res = requests.post(f'{base_url}/Login/{usuario},{contraseña}')
45     data = res.text#convertimos la respuesta en dict
46     messagebox.showinfo("LOGIN",data)
47     print(data)
48 def editN(nombre):
49     res = requests.post(f'{base_url}/editN/{nombre}')

```

