Universidad San Carlos de Guatemala
Estructura de Datos

Manual Técnico

Proyecto
Fase 2

Nataly Saraí Guzmán Duarte

Carnet: 202001570

Contenido

Introducción 3
Información destacada 3
Requerimientos 4
Datos técnicos para la realización del sistema 4
..... 5
Lógica del programa 5
..... 8
..... 9
Diagrama general de la aplicación 9
Conclusion 10

Introducción

En el presente documento se adjuntan las caracteristicas, funcionamiento y estructura que fueron requeridas para el desarrollo del sistema presentado, el cual consta de un sistema de carga de informacion como varias funcionalidades de usuarios, tambien cuenta con un sistema de juego utilizando estructuras de datos para que estas sean lo mas eficiente posible.

Lugar de realización: Guatemala

Fecha:03/10/2022

Responsable de elaboración: Nataly Saraí Guzmán Duarte

Información destacada

El juego Batalla naval es un juego que tiene como raiz el juego busca minas, con los mismos principios. Con la informacion de lo que deseamos alcanzar podemos establecer algunas formas para su desarrollo, en este caso nos es preferible utilizar estructuras de datos para el manejo de la informacion.

Requerimientos

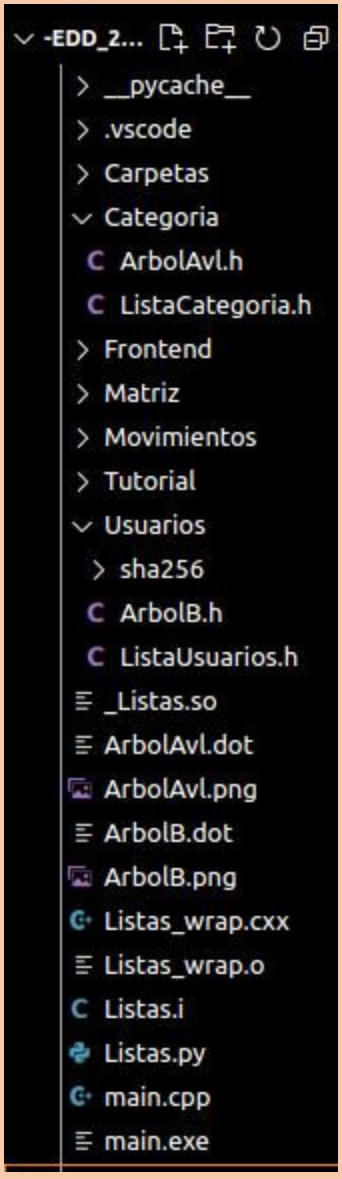
- RAM: 1 GB (mínimo).
- ROM: 250 MB (mínimo).
- Arquitectura x32 bits o x64 bits.
- Sistema operativo: Windows, Linux, MacOS.
- Lenguaje de programación: python,c++.
- Plataforma IDE: Visual Estudio Code

Datos técnicos para la realización del sistema

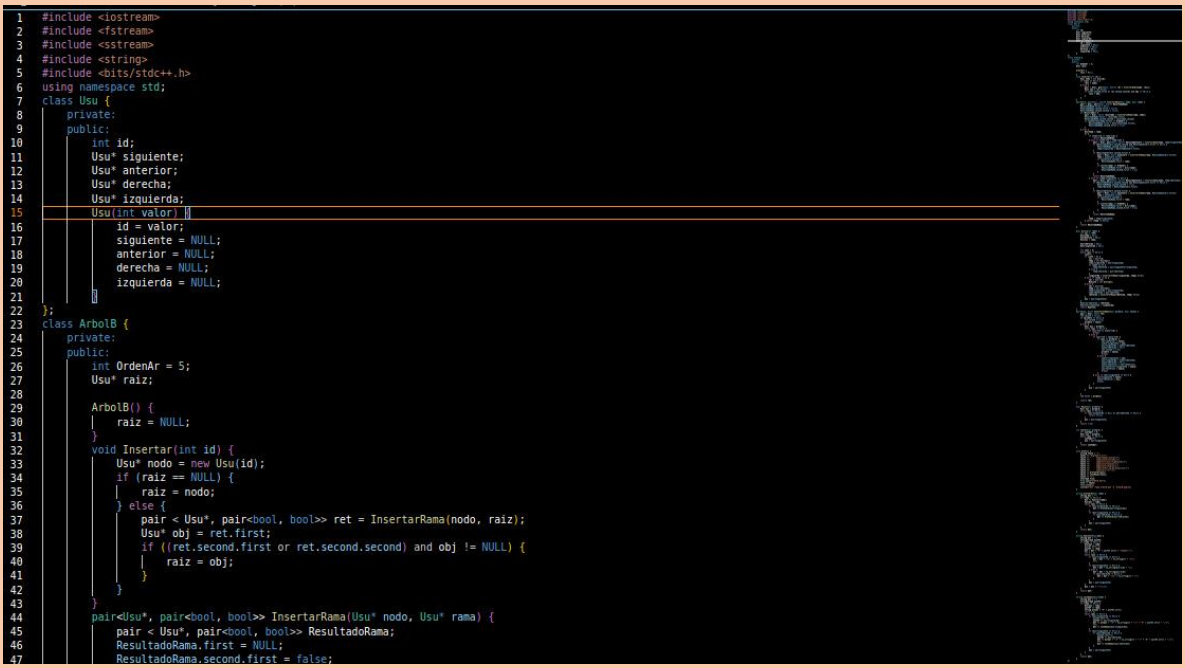
- Ubuntu 22.04
- Procesador: Intel Core i5.
- RAM: 8.00 GB
- Plataforma IDE: Visual Estudio Code

Lógica del programa

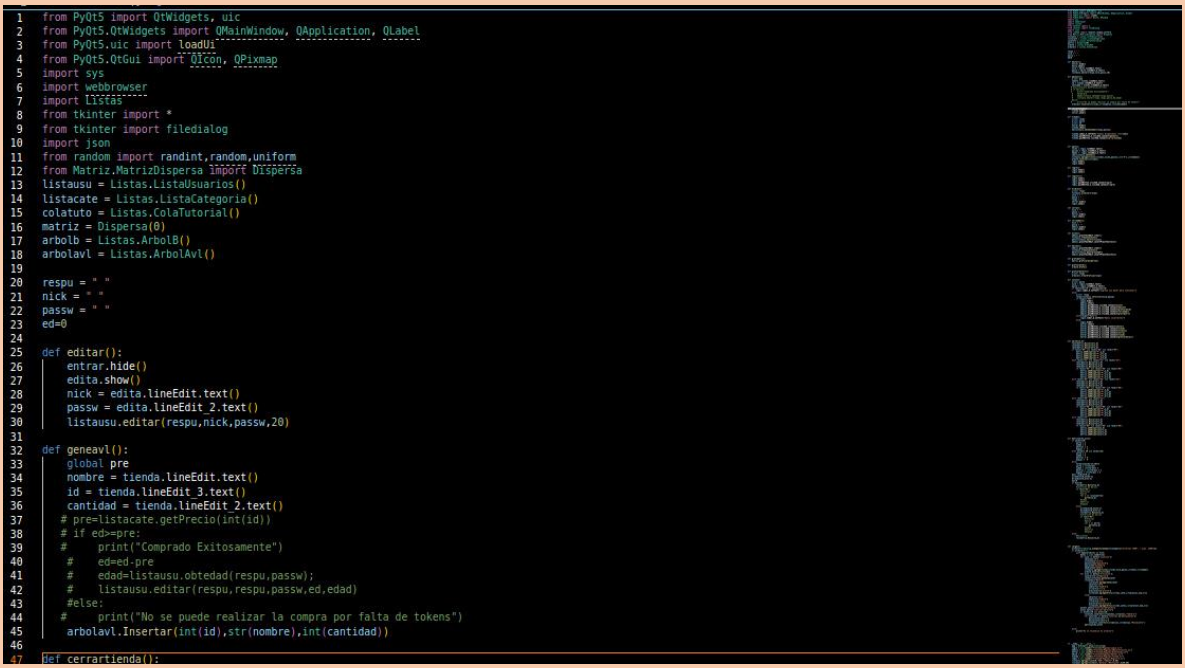
Se realizo una distribucion del sistema de la siguiente forma tratando de tener el mayor orden posible para posteriores mejoras en el sistema, donde podemos ver las diferentes agrupaciones que se realizan para cada estructura aplicada.



Dentro de estos achivos podemos observar algunas secciones de codigo que son de vital importancia para el flujo de la informacion, en este caso podemos ver que para guardar los datos de los usuarios lo realizaremos en un arbol B de nivel 5.

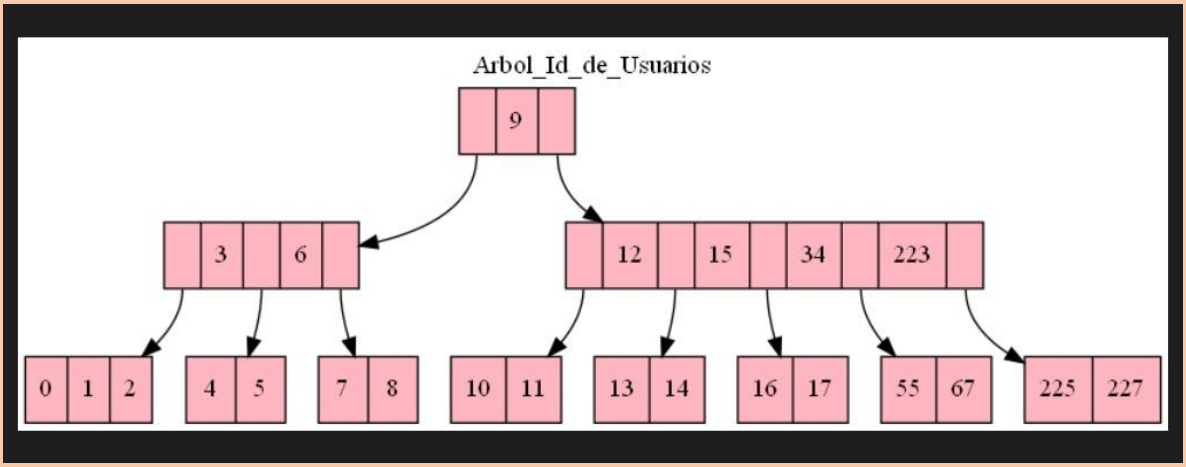


```cpp
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <bits/stdc++.h>
using namespace std;
class Usu {
    private:
    public:
        int id;
        Usu* siguiente;
        Usu* anterior;
        Usu* derecha;
        Usu* izquierda;
        Usu(int valor) {
            id = valor;
            siguiente = NULL;
            anterior = NULL;
            derecha = NULL;
            izquierda = NULL;
        }
};
class ArbolB {
    private:
    public:
        int OrdenAr = 5;
        Usu* raiz;

        ArbolB() {
            raiz = NULL;
        }
        void Insertar(int id) {
            Usu* nodo = new Usu(id);
            if (raiz == NULL) {
                raiz = nodo;
            } else {
                pair < Usu*, pair<bool, bool>> ret = InsertarRama(nodo, raiz);
                Usu* obj = ret.first;
                if ((ret.second.first or ret.second.second) and obj != NULL) {
                    raiz = obj;
                }
            }
        }
        pair<Usu*, pair<bool, bool>> InsertarRama(Usu* nodo, Usu* rama) {
            pair < Usu*, pair<bool, bool>> ResultadoRama;
            ResultadoRama.first = NULL;
            ResultadoRama.second.first = false;
```



```python
from PyQt5 import QtWidgets, uic
from PyQt5.QtWidgets import QMainWindow, QApplication, QLabel
from PyQt5.uic import loadUi
from PyQt5.QtGui import QIcon, QPixmap
import sys
import webbrowser
import Listas
from tkinter import *
from tkinter import filedialog
import json
from random import randint,random,uniform
from Matriz.MatrizDispersa import Dispersa
listausu = Listas.ListaUsuarios()
listacate = Listas.ListaCategoria()
colatuto = Listas.ColaTutorial()
matriz = Dispersa(0)
arbolb = Listas.ArbolB()
arbolavl = Listas.ArbolAvl()

respu = " "
nick = " "
passw = " "
ed=0

def editar():
    entrar.hide()
    edita.show()
    nick = edita.lineEdit.text()
    passw = edita.lineEdit_2.text()
    listausu.editar(respu,nick,passw,20)

def geneavl():
    global pre
    nombre = tienda.lineEdit.text()
    id = tienda.lineEdit_3.text()
    cantidad = tienda.lineEdit_2.text()
   # pre=listacate.getPrecio(int(id))
   # if ed>=pre:
   #     print("Comprado Exitosamente")
    #    ed=ed-pre
    #    edad=listausu.obtedad(respu,passw);
    #    listausu.editar(respu,respu,passw,ed,edad)
    #else:
   #     print("No se puede realizar la compra por falta de tokens")
    arbolavl.Insertar(int(id),str(nombre),int(cantidad))

def cerrartienda():
```

El usuario podra tambien realizar compra de skins para mejorar la interfaz que tendra en el momento de generar un nuevo juego, asi tambien podra tener mas de una skin, con ello podemos notar que la mejor forma de guardar estas skins es en un arbol avl.



Para el apartado de juego podemos generar una matriz dispersa ya que con ello podemos ahorrarnos espacios de memoria y simplificar el sistema de validacion de existencia de las ubicaciones de los nodos que representen a los barcos y tambien utilizaremos lista de cabeceras.

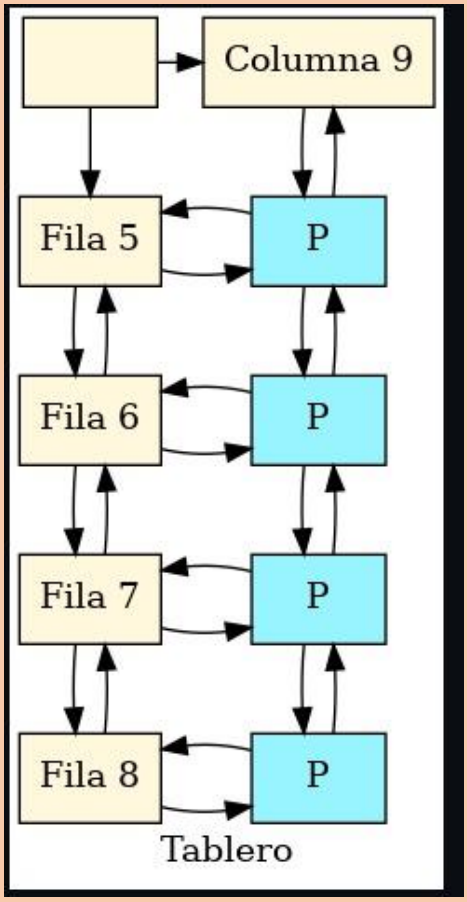
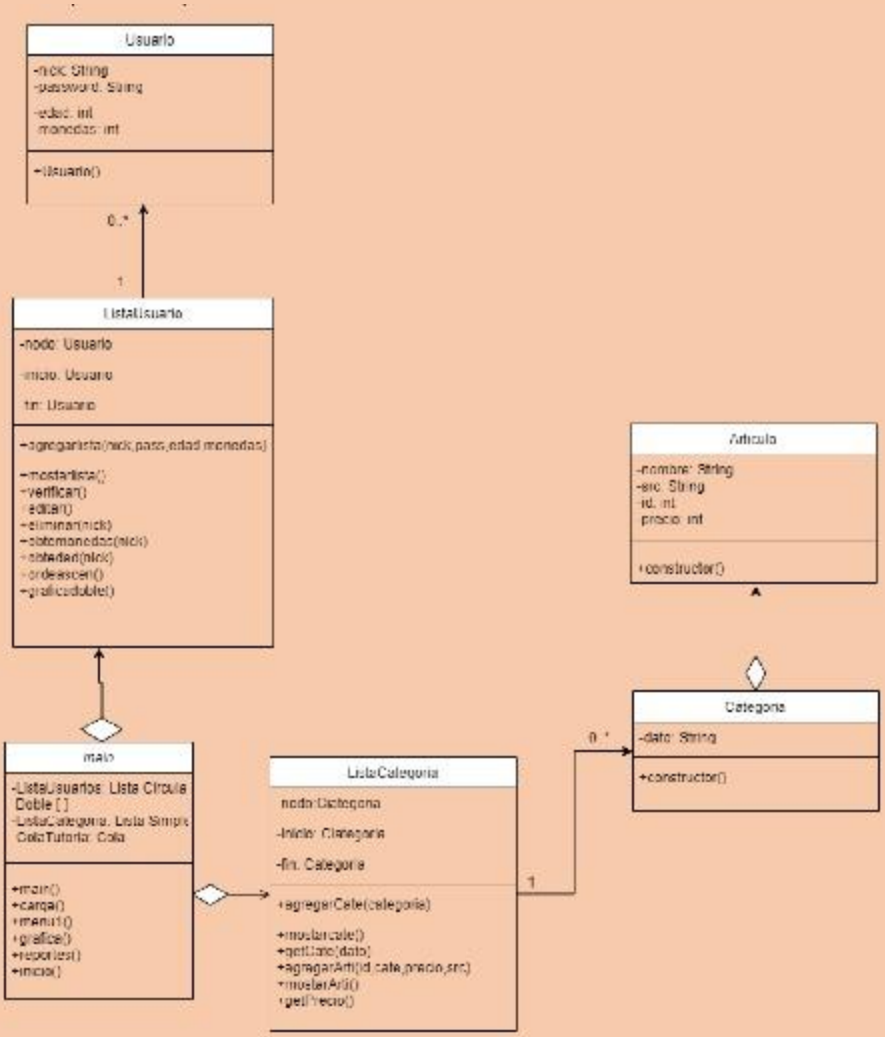


Diagrama general de la aplicación

Muestra el sistema de organización que se utilizo para desarrollar este sistema en una estructura UML de clases con ello podemos validar la facil adaptacion en caso de ser requerido a un nuevo sistema de lenguaje.



Conclusion

Uno de los problemas que tenemos al programar es la eficiencia, y esto en dos aspectos, uso de la memoria y rapidez de acceso a la informacion, en este proyecto podemos poner en práctica algunos aspectos que nos son de utilidad para poder afrontar los retos que nuestra carrera nos presenta.

Cuando observamos como se conforma el sistema de ingreso de usuarios que en su defecto es un arbol B, al compararlo con la fase anterior podemos notar una diferencia en las respuestas de ambas estructuras, siendo de menor tiempo de respuesta el arbol B asi al aplicarlo dentro de un sistema tenemos mas eficiencia en la busqueda de datos siempre teniendo en cuenta el uso de memoria.