



Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Estructura de Datos Sección
Catedrático: Ing. Álvaro Hernández
Tutor académico: Alex López

SECCION B

Segundo semestre 2022

MANUAL TÉCNICO

Proyecto 1 Fase 3

USAC Games, Batalla naval

Nombre: Moises David Maldonado de León

Carné: 202010833

Guatemala, 21 de octubre de 2022

Introducción

En el presente documento, se describe la estructura y los demás aspectos técnicos de la fase 1,2 y 3 del proyecto USAC Games, el cual es desarrollar una aplicación del conocido juego de Batalla Naval, así como servicio de transacciones respaldado con tecnología Blockchain y almacenamiento de monedas en un wallet, así como la generación y visualización dinámica de reportes. Se conocerá a detalle de las estructuras implementadas, así como de sus respectivos métodos y funciones para estas fases.

OBJETIVOS

Generales

Familiarizar al lector con la lógica planteada para el funcionamiento del sistema realizado mediante el uso del lenguaje de programación C++ y Python aplicando los conocimientos sobre estructura de datos para la parte del manejo de información, Blockchain y wallet para el manejo de transacciones, así como la implementación de la matriz dispersa y generación de gráficos mediante Graphviz y jugabilidad para 1 y 2 jugadores.

Específicos

- Mostrar la estructura del programa realizado.
- Dar a conocer los métodos y funciones empleados, así como su funcionalidad.
- Mostrar el diagrama de clases implementado para esta fase.

Alcances

El programa se realizó con el fin de implementar un videojuego que permitan desarrollar la agilidad mental de los usuarios, específicamente con el conocido juego de Batalla Naval, sin embargo se espera una aplicación compleja, por lo que el alcance se maximiza debido a que se implementará un apartado visual amigable e intuitivo para el usuario así como la posibilidad de crear cuenta, obtener puntos y poder comprar en el apartado de tienda así como una funcionalidad de juego que le permita desarrollar su agilidad mental. También cuenta con seguridad para las transacciones monetarias.

Especificaciones

Requisitos del Hardware

- Mínimo 500 MB de memoria RAM
- Procesador Intel Core 2 Duo 2 o superior
- Espacio en disco duro: 70 MB

Requisitos del Software

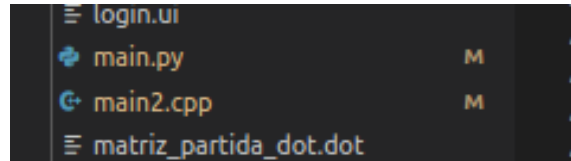
- El programa fue realizado en C++ y Python
- El IDE utilizado fue Visual Studio Code
- Funcional en sistema operativo Ubuntu
- Tener instaladas o con acceso a las siguientes librerías y paquetes: Graphviz, Crow.

Lógica del programa

Descripción de clases		
Archivo	Clase	Descripción
ColaTutorial.h	Cola	Maneja la estructura de cola. Contiene las llamadas a métodos de la clase.
NodoTutorial.h	nodoTutorial	Contiene los atributos del nodo y llamadas de métodos. Se trabajó como lista doble.
ListaArticulos.h	listaArticulos	Maneja la estructura de lista simple. Contiene apuntador inicio, así como las llamadas a métodos de la clase.
NodoArticulos.h	nodoArticulos	Contiene los atributos del nodo, así como su apuntador y llamadas de métodos.
ListaCategoria.h	listaCategoria	Maneja la estructura de lista simple. Contiene apuntador inicio, así como las llamadas a métodos de la clase.
NodoCategoria.h	nodoCategoria	Contiene atributos del nodo, además un atributo de tipo listaArticulos para manejar la estructura de lista de listas.
ListaJugadas.h	listaJugadas	Maneja la estructura de lista simple. Contiene apuntador inicio, así como las llamadas a métodos de la clase.
NodoJugada.h	nodoJugada	Atributos del nodo, uno es de tipo pila para manejar la estructura de lista de pilas.
ListaUsuarios.h	listaUsuarios	Maneja la estructura de lista doble circular. Contiene apuntadores, así como las llamadas a métodos de la clase.
NodoUsuario.h	Usuario	Contiene atributos del nodo, además un atributo de tipo listaJugadas para manejar la estructura de lista de pilas.
PilaMovimientos.h	Pila	Maneja la estructura de pila. Contiene apuntadores, así como las llamadas a métodos de la clase.
NodoMovimiento.h	NodoDisparo	Contiene atributos del nodo y llamada de métodos de la clase. Se trabajó como lista doble.
arbolAVL.h	Compra AVL	Clase compra será el nodo donde se guardará la información. Clase árbol que maneja la búsqueda e inserción
arbolB.h	ArbolB	Clase árbol que maneja la búsqueda e inserción dentro del árbol.
NodoUserAB	usuarioB	Clase nodo que guarda la información de los usuarios.
ListaEncabezado.py	ListaEncabezado	Clase que maneja los encabezados de la matriz dispersa, tanto para filas y columnas
Main.py	Mt tutorialM	Ambas clases almacenan una matriz dispersa, la primera es para el tablero de juego y la segunda es para el tutorial
Matriz.py	Nodo intero Matriz dispersa	La primera clase almacena la información del nodo interno (contenido, fila y columna). La segunda clase maneja la inserción y demás algoritmos para el manejo de la matriz dispersa
NodoEncabezado.py	Nodo encabezado	Contiene atributos del nodo encabezado, su posición y apuntadores
StoreC.py	MainWindow	Clase que maneja la interfaz de tienda para el llenado y mostrado de datos
tablaHASH.py	tbHash	Clase que almacena la tabla hash para el carrito de compras

Archivo main2.cpp/main.py

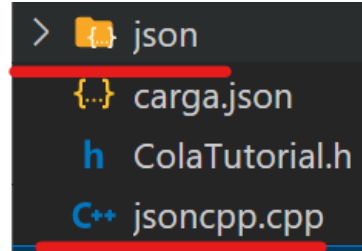
“main2.cpp” funciona de “Servidor”.
El archivo “main.py” maneja la parte de la interfaz gráfica, así como el uso de la matriz dispersa.



```
login.ui
main.py
main2.cpp
matriz_partida_dot.dot
```

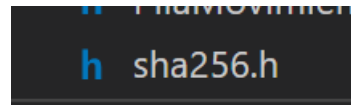
Archivos complementarios

La caperta json y el archivo llamado “jsoncpp.cpp” permiten la correcta lectura de archivos json así como su manipulación para el manejo de carga masiva.



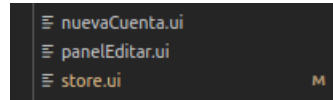
```
> json
{...} carga.json
h ColaTutorial.h
C++ jsoncpp.cpp
```

El archivo llamado “sha256.h” permite convertir en un hash la contraseña y así mantener la seguridad del usuario.



```
h sha256.h
```

Los archivos .ui son creados por Qt 5 Designer el cual se usó para diseñar las ventanas de la interfaz.



```
nuevaCuenta.ui
panelEditor.ui
store.ui
```

Descripción de las funciones y métodos

Clase	Método/Función	Descripción
Cola{}	cola()	Constructor de la lista, inicializa los apuntadores en NULL.
	Void queue(string&,string&,string&,string&)	Método para la inserción al inicio de la cola. Se deben de enviarle los parámetros solicitados.
	string mostrarTutorial()	Método que devuelve un JSON con los datos del tutorial
	Void dequeue()	Método que elimina el primer nodo insertado
	Void generarReporte()	Este método genera una imagen de tipo png donde se visualiza la estructura de la cola y su información.
listaArticulos{}	void insertarInicio (string&,string&,string&,string&)	Método para la inserción al inicio de la lista. Se deben de enviarle los parámetros solicitados.
	Bool verificar ID() String getArticulos()	Función que verifica que el ID exista en la lista. Método que retorna un JSON con los datos de 1 artículo
	String getName String getPrice	Funciones que retornar el nombre y el precio de un artículo en específico.
listaCategoria{}	Void insertarInicio(string&);	Método para la inserción al inicio de la lista. Se deben de enviarle los parámetros solicitados.
	String getNombre String getPrecio	Funciones que retornan el precio y el nombre de un articulo en una categoría en específico
	String getDatos()	Método que retorna un JSON con todos los artículos de la tienda
	Void insertarNuevoArticulo (string&, string&,string&,string&,string&)	Método para la inserción de un nuevo articulo. Busca por nombre de categoría e inserta el nodo artículo.
	bool verificarExistencia(string&); bool verificarID()	Funciones para validar la existencia de un articulo y la categoría

Descripción de las Métodos/funciones		
Clase	Método/función	Descripción
listaUsuarios{}	Void insertarNuevo (string&,string&,string&,string&)	Método para la inserción al final en la lista circular doblemente enlazada de un nuevo usuario.
	Void insertarJugada (string&,string&,string&);	Método para la inserción al principio en la lista simple de jugadas según sea el usuario que tenga la sesión iniciada.
	Void nuevoMovimiento (string&,string&,string&,string&,string&);	Método para la inserción de nuevo movimiento en la pila de movimientos. Esta llama a un método de la lista de jugadas para nuevo movimiento.
	Void ReporteUsuarios();	Genera una imagen de tipo png donde se visualiza la lista de usuarios.
	int login(string&,string&);	Función que retorna 1 si se encuentra el usuario (y le permite acceder al menú de usuario) y 0 si no.
	Void modificarInformacion(string&,string&);	Método que despliega un menú y permite modificar el nombre, contraseña o edad.
	Void sumarPunto(string&,string&);	Método de busca el usuario y luego suma 1 token/moneda en su registro.
	bool verificarNombre(string&);	Función que retorna verdadero si el nombre existe y falso si no. Este método se llama en la inserción de nuevo usuario y modificar nombre como validación.
	Void reporteJugadas();	Genera una imagen de tipo png donde se visualiza la lista de pilas de cada usuario.
	Void eliminarCuenta(string&,string&);	Función para eliminar la cuenta del usuario.
	string getMonedas string getID	Función que devuelve una cadena de texto con valor de las monedas que dispone el usuario. Este método se llama en sumarPunto(); método que retorna el ID del usuario.
	Void GraficarARbol()	Grafica el árbol B generando una imagen PNG
	insertarAB() ModificarAB()	Métodos para inserción y modificación en el árbol B
	Int verAVL() Int nuevaCompra()	Funciones para la inserción de nodos en el AVL así como la generación del gráfico PNG
	Void restarMonedas()	Metodo que recibe una cantidad de monedas y la resta a la cantidad actual.
	String verUltimoMovimiento() Void eliminarUltimoMovimiento()	Función que retorna un JSON con la información del ultimo movimiento. Metodo que elimina el ultimo movimiento en la pila
	Void Reinserción()	Metodo que genera un nuevo árbol si se elimina un usuario de la lista.

Descripción de métodos/funciones		
Clase	Método/función	Descripción
pila{}	Void push(string&,string&);	Método que inserta una nueva jugada en la pila de movimientos del usuario.
	string mostrarPila (int&,int&,int&);	Función que retorna una cadena de texto de nodos y sus uniones con la información de la pila. Este método se llama desde lista de jugadas.

Descripción de métodos/funciones		
Clase	Método/función	Descripción
listaJugadas{}	Void nuevaJugada (string& nombre);	Método que inserta una nueva jugada en la lista de jugadas del usuario, esta nueva jugada guardará una pila de movimientos.
	string verTop() void eliminarTop()	Función que solicitan a la pila el ultimo dato apilado y método de eliminación del mismo.
	void nuevoMovimiento (string& ,string&,string&);	Función para la inserción de un nuevo movimiento. Recibe el nombre de jugada, una vez la encuentra procede a la inserción en la Pila de movimientos.

Descripción de métodos/funciones		
Archivo/clase	Método/función	Descripción
Mt/tutorialIM	_init_	Inicializa la matriz y contiene el nombre de la jugada
	putSize()	Metodo que asigna el tamaño de la matriz
	Insert()	Metodo que inserta nodos en la matriz dispersa
	generarPosicionesAleatorias()	Metodo que genera los barcos de manera aleatoria en el tablero
	graficarNeato()	Metodo que grafica la matriz dispersa
	insertarMovimiento()	Metodo que inserta el movimiento del jugador y determina si acertó o no.
	verificarGanar()	Recorre la matriz para verificar si todos los barcos están destruidos.
	verificarAciertos()	Metodo que genera la cantidad de naves destruidas.
	Reestablecer()	Metodo que reestablece el tablero a su estado anterior a un movimiento
	Eliminar()	Metodo que elimina la matriz para poder reutilizarla

Descripción de métodos/funciones		
Clase	Método/función	Descripción
listaJugadas{}	Void nuevaJugada (string& nombre);	Método que inserta una nueva jugada en la lista de jugadas del usuario, esta nueva jugada guardará una pila de movimientos.
	string verTop() void eliminarTop()	Función que solicitan a la pila el ultimo dato apilado y método de eliminación del mismo.
	void nuevoMovimiento (string& ,string&,string&);	Función para la inserción de un nuevo movimiento. Recibe el nombre de jugada, una vez la encuentra procede a la inserción en la Pila de movimientos.

Descripción de las Métodos/funciones		
Clase	Método/función	Descripción
main.py	loginAction()	Obtiene los datos del usuario, realiza la petición y realiza las validaciones necesarias
	eliminarCuenta()	Realiza la petición al servidor para eliminar la cuenta, vuelve al login luego de ejecutarse
	verUsuario()	Método que despliega la ventana del usuario
	nuevaCuenta()	Despliega la ventana de registro
	getDatosNuevaCuenta()	Obtiene los datos del registro y realiza la petición para la inserción en lista y árbol.
	updateData()	Obtiene los datos de actualización y los envía en una petición al servidor.
	volverEditar() editarDatos()	Funciones que permiten regresar al menú de usuarios o ir a la ventana de actualización respectivamente
	logOut() logOutAdmin()	Funciones para cerrar la sesión tanto para el usuario como para el administrador.
	cargaMasiva()	Realiza la petición para la carga masiva y obtiene la información de los artículos para llenar la ventana de tienda.
	irTienda() backTienda()	Función que muestra la interfaz de tienda o función que permite regresar.
	salir()	Función que termina con la ejecución del programa
	Pagar() verCompras()	Función que permite realizar compras al usuario y función que despliega una imagen PNG con sus respectivas compras.
	getCoins() setCoins()	Métodos para obtención y modificación de monedas de 1 usuario
	Prueba() getIDUser()	Función que solicita el tamaño del tablero, nombre de jugada y llama a la generación, grafica y mostrado del tablero; Función que obtiene el id del usuario actual.
	generarMatriz() generarMatrizPrincipal() generarMatrizInvitado()	Métodos que generan el tablero para cada modalidad de juego.
	getImage() getImageInvitado() getImageBack()	Método que muestra el tablero o la imagen principal del juego tanto para 1 jugador como para jugador vs jugador.
	makeMove() makeMoveInv deshacer()	Método que permite obtener el movimiento del jugador, le indica si acertó o no, así como hace las validaciones de ganar partida o perderla, así como si desea jugar de nuevo con la configuración previamente establecida. El método de deshacer permite reestablecer el tablero a un movimiento anterior y le quita 5 puntos (solo disponible para el jugador principal).
	resetGame() resetGameJ1() resetGameJ2()	Elimina la matriz para los casos de 1 jugador, o jugador vs jugador
	getTutorial	Método que obtiene un JSON del servidor con la información del tutorial y utiliza una "animación" para mostrar todos los movimientos. Al terminar se cierra la ventana
	showTutorial()	Muestra la ventana del tutorial
	getImageTutorial()	Muestra el tablero del tutorial
	generarMatrizTutorial()	Genera el tablero inicial del tutorial
	elegirPosiciones()	Método para colocar manualmente los barcos del jugador principal
	viewCar() – backCar()	Métodos para el mostrado y ocultamiento de la ventana del carrito de compras.
	sendCar() – eliminarFila()	Métodos para añadir productos a la lista de carrito o eliminarlos.
	graphCar() – cancelar()	Método para graficar el carrito de compras así como método para vaciar el carrito

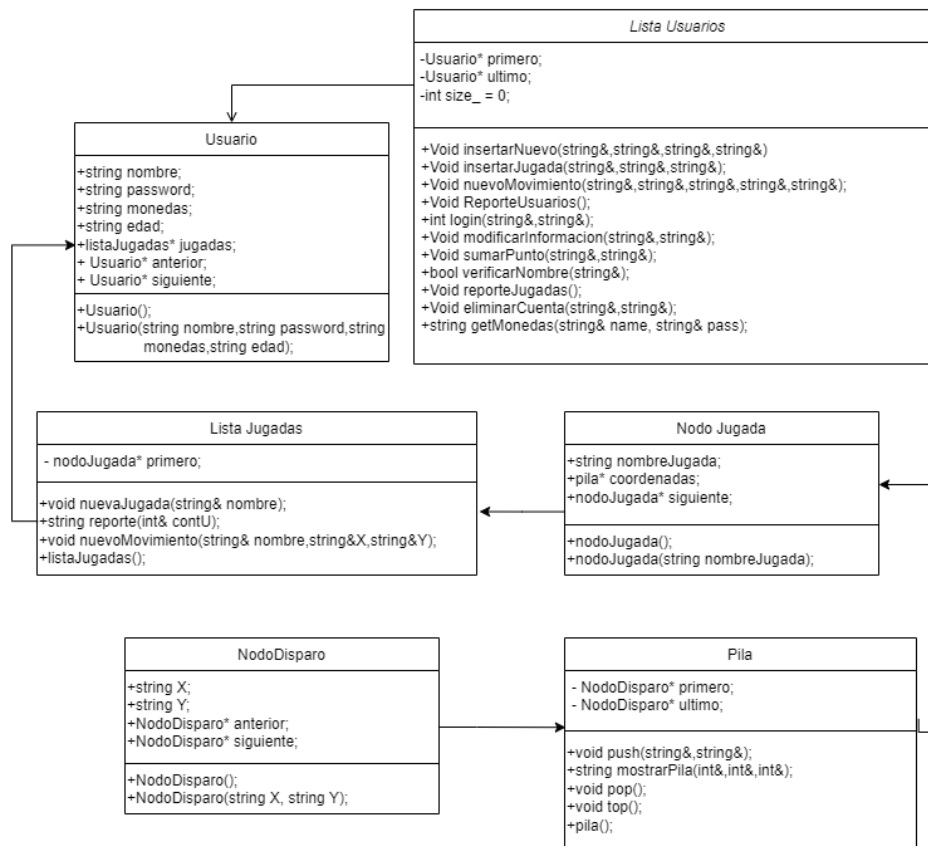
Descripción de las Métodos/funciones		
Clase	Método/función	Descripción
Matriz Dispersa()	_init_()	Función que inicializa la matriz, inicia la capa, las listas de encabezados (Filas y columnas), contadores y guardado de movimientos.
	putSize()	Método para indicar el tamaño de la matriz.
	insertar()	Método para insertar nodos celda dentro de la matriz.
	ubicarPortavion(x,y) ubicarSubmarino(x,y) ubicarDestructor(x,y) ubicarBuque(x,y)	Métodos que reciben las coordenadas de fila y columna, ubican la coordenada y proceden a recorrer los caminos disponibles (izquierda,derecha,arriba,abajo) guardando las coordenadas en 4 arreglos distintos. Una vez se haya completado la extensión de cada nave (4,3,2,1 respectivamente) se llama a los métodos de pintar.
	pintarPortavion(x,y) pintarSubmarino(x,y) pintarDestructor(x,y) pintarBuque(x,y)	Métodos que reciben las coordenadas de fila y columna, ubican la coordenada y proceden cambiar el carácter de la celda según sea la letra de la nave.
	getShips()	Función que retorna un json con la cantidad de cada barco a partir de la ecuación planteada dentro de la misma.
	generarPosicionesAleatorias()	Método que genera las posiciones aleatorias de los barcos según sea la configuración establecida.
	getFila() getColumna()	Funciones que retornan un valor aleatorio en el rango de la cantidad de filas o columnas para la fila y columna respectivamente.
	graficarNeato()	Método que genera una imagen png con el tablero, según sea el contenido de la celda se mostrarán de distintos colores.
	verificarPartidaGanada()	Función que recorre la matriz, si todas las celdas no contienen una letra identificadora de cualquier nave entonces retorna un mensaje indicando que se ha ganado.
	reestablecer()	Método que reestablece el ultimo movimiento hecho en el tablero.
	verificarAciertos()	Método que actualiza los contadores de los barcos destruidos al finalizar una partida.
	verListaAdyacencia()	Método que genera una imagen png con la lista de adyacencia a partir de los encabezados de filas de la matriz dispersa.
	grafoAdyacencia()	Método que genera una imagen png con el grafo que representa la lista de adyacencia a partir de los encabezados de filas de la matriz dispersa.
	generarMatriz()	Metodo que genera el tablero
	getImage() getImageBack()	Metodo que muestra el tablero o la imagen principal del juego
	makeMove() deshacer()	Metodo que permite obtener el movimiento del jugador, le indica si acertó o no así como hace las validaciones de ganar partida o perderla así como si desea jugar de nuevo con la configuración previamente establecida. El metodo de deshacer permite reestablecer el tablero a un movimiento anterior y le quita 5 puntos.
	resetGame()	Elimina la matriz
	getTutorial	Metodo que obtiene un JSON del servidor con la información del tutorial y utiliza una "animación" para mostrar todos los movimientos. Al terminar se cierra la ventana
	showTutorial()	Muestra la ventana del tutorial
	getImageTutorial()	Muestra el tablero del tutorial
	generarMatrizTutorial()	Genera el tablero inicial del tutorial

Descripción de las Métodos/funciones		
Clase	Método/función	Descripción
tbHash	_init_()	Función que inicializa el contador de ocupación, así como la tabla (arreglo) y su método para el tamaño inicial.
	tamañoInicial()	Método para llenar la tabla de valores None según el tamaño inicial establecido.
	insertar()	Método para insertar un valor en la tabla hash, recibe un arreglo con los datos de id del usuario y producto, nombre del producto y su respectivo precio
	getKey()	Método para obtener un índice a partir de la llave (Id usuario concatenado con nombre del producto)
	llaveExistente()	Función que verifica si existe una llave en el índice indicado, retorna true o false
	Colision()	Función que crea un nuevo índice a partir de la función establecida dentro de la misma.
	Rehash()	Función ejecutada al 80% de la capacidad, determina el numero de posiciones que será la nueva tabla y muda los datos de la tabla actual a la nueva tabla de tal modo que el porcentaje de ocupación sea el 20% en la nueva tabla.
	verTabla()	Método de prueba que imprime en consola la tabla.
	getJson()	Función que retorna una cadena string con estructura json de los datos de la tabla.
	getGraph()	Método que grafica la tabla hash.
	vaciarTabla()	Método que elimina el contenido de la tabla hash y regresa al tamaño inicial
	eliminarDato()	Método para eliminar un dato en específico de la tabla hash.

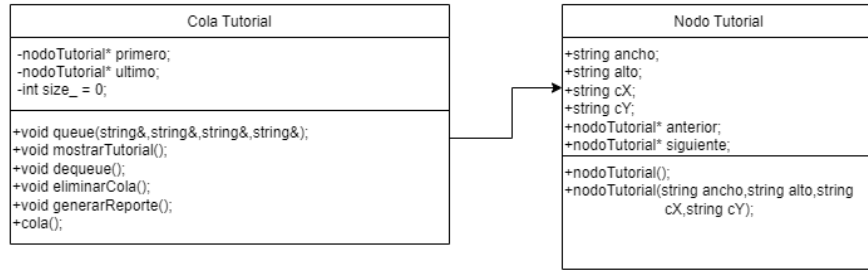
Flujo del programa

Diagrama de clases (Fase 1)

Usuarios



Tutorial



Artículos

