Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y sistemas
Estructuras de Datos
Ingenieros:

- Ing. Luis Espino
- Ing. Edgar Ornelis
- Ing. Álvaro Hernández

Auxiliares:

- Alex Lopez
- Walter Mach
- Wilfred Perez

Proyecto 1 Fase 2

USAC Games, Batalla naval

Índice

Objetivos

Objetivo general

- Aplicar los conocimientos del curso Estructuras de Datos en el desarrollo de una aplicación que permita manipular la información de forma óptima.

Objetivos específicos

- Demostrar los conocimientos adquiridos sobre estructuras de datos no lineales: matrices y arboles, poniéndolos en práctica en el desarrollo del juego batalla naval.
- Utilizar el lenguaje C++ para implementar estructuras de datos no lineales.
- Utilizar el lenguaje de programación Python para el desarrollo de interfaces gráficas.
- Utilizar la herramienta Graphviz para graficar estructuras de datos lineales.
- Definir e implementar algoritmos de búsqueda, recorrido y eliminación.

Descripción General del Proyecto

La empresa Usac Games desea implementar un videojuego que permitan desarrollar la agilidad mental de los usuarios, por lo cual ha planeado desarrollar una aplicación con el juego Batalla naval y le solicita a usted como estudiante de estructura de datos poder implementar algoritmos, funciones y estructuras que permitan que el juego tenga un rendimiento óptimo y fluido.

Debido al alcance y complejidad de las funcionalidades del videojuego se ha decidido dividirlo en 3 fases. A continuación se describen de forma general las funcionalidades a cubrir en cada una de las fases.

Fase 1 (20 pts - C++)

- **Listas**: Registro de usuarios
    - **Encriptación**: Aplicar seguridad a la información de los usuarios
- **Lista de listas**: Tienda de Skins del juego (los puntos por partida serán la moneda)
- **Pila**: Retroceder jugadas (Push y pop de movimientos)
- **Cola**: Tutorial del juego (push y pop de información con movimientos)

Fase 2 (30 pts - C++ y Python)

- **Matrices**: Tablero del juego y eventos para realizar movimientos
- **Árbol B**: Indexación de usuarios por orden alfabético
- **Árbol AVL**: Compras del usuario ordenadas por precio.

Fase 3 (40 pts - C++ y Python)

- **Tabla Hash:** Bitácora de eventos del sistema
    - Registro de usuarios, skins, compras de los usuarios y jugadas realizadas
- **Grafos**: Lista adyacente y grafo de movimientos por partida del usuario
- **BlockChain**: Transacciones al realizar compras

Consideraciones

1. Las funcionalidades están descritas de forma general, las mismas pueden sufrir algunas modificaciones en el enunciado final de cada fase.
2. La funcionalidad correcta del videojuego será cubierto por las 3 fases, por lo que de no realizar una de las fases, supondrá un mayor esfuerzo por parte del estudiante para completar la fase siguiente, el desarrollo de las estructuras se realizará en C++ y la funcionalidad visual se realizará en Python.

Fase 2

En esta fase del proyecto, Usac Games desea que se pueda implementar el juego totalmente funcional, haciendo uso de las estructuras codificadas en la fase 1 e integrando una interfaz gráfica de usuario.

Inicio de sesión

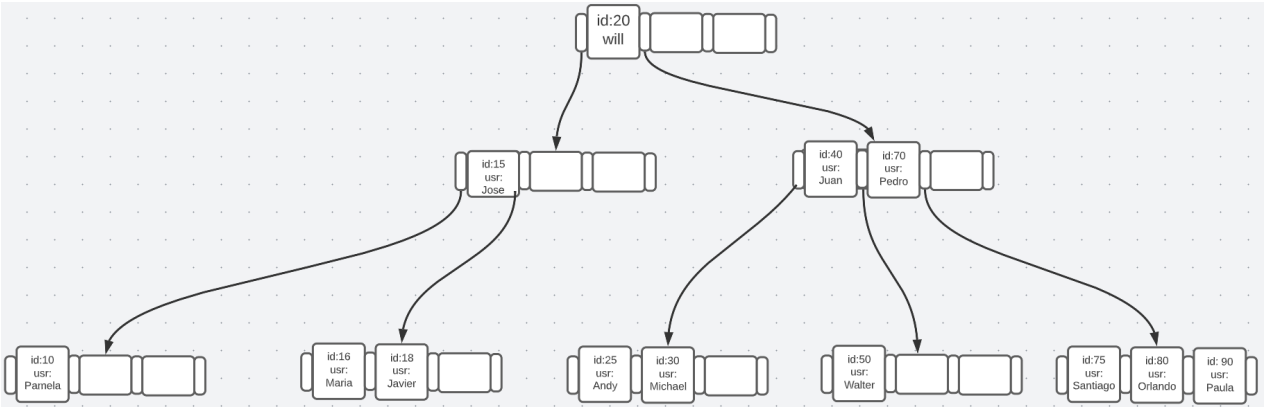
La aplicación contará con un inicio de sesión, validará que el usuario y contraseña sean correctos para poder ingresar, de lo contrario deberá mostrar un mensaje de error.

Operaciones con los usuarios:

- Carga de usuarios: Se utilizará el archivo de entrada.
- Edición de usuarios: Se tendrá un apartado con el que cada usuario modificara su información
- Eliminación de usuarios: Se podrán eliminar a un usuario si se desea, esto eliminará toda su información incluyendo monedas y skins adquiridas por el usuario.

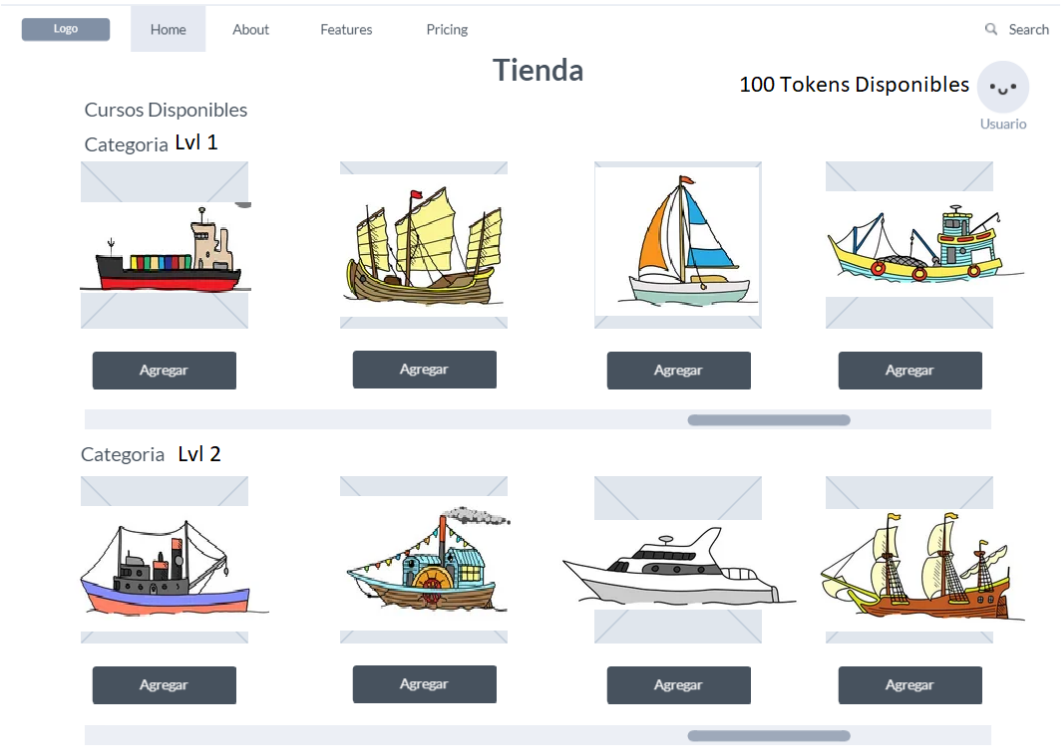
Almacenamiento de usuarios

El almacenamiento de los usuarios se añadirá a un arbol B indexador por su identificador único, añadiendo mayor rendimiento a la aplicación al momento de buscar un usuario.
Link de la imagen:
https://drive.google.com/file/d/13NwWF03PnsVaW6cDB2MYAMLbXZNtB8Yr/view?usp=sharing

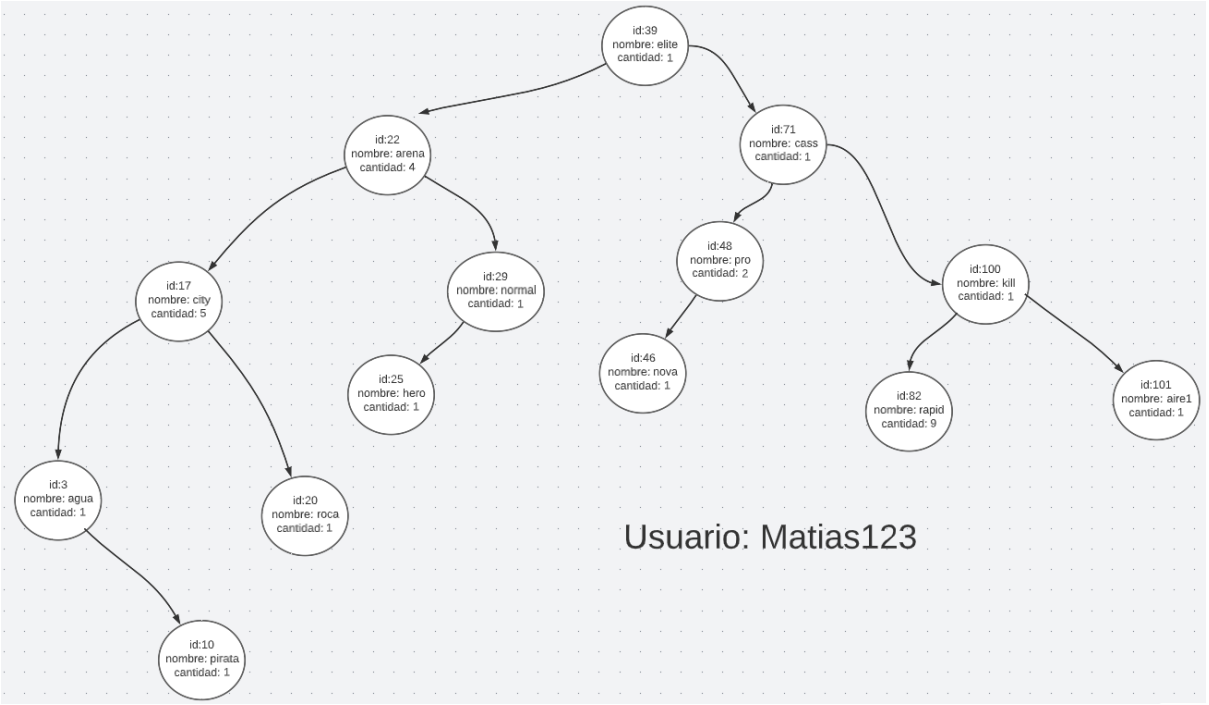


Tienda de Skins

En este apartado se tendrá que mostrar un mosaico con todas las texturas disponibles para los distintos barcos, cada usuario tendrá una cantidad de tokens con los cuales podrá comprar las diferentes skins, estos tokens no son ilimitados, deberá disminuir el número de tokens según la compra realizada.



El usuario almacenará todas las compras del juego en un árbol avl, se utilizará el id del artículo para la indexación del árbol.
Link:
https://drive.google.com/file/d/1cuF8I1qtO4aul6U9mxnfFROcf7xCqniD/view?usp=sharing



Componentes del juego

- Tablero de m x m posiciones
    - para m ≥ 10
- Barcos
    - Portaaviones (4 x 1) ó (1 x 4).
    - Submarino(3x1) ó (1x3).
    - Destructor (2x1) ó (1x2).
    - Buque (1x1)

Lógica del juego

Construcción del tablero

Se deben de utilizar los barcos disponibles (portaaviones, submarino, destructor, buque). Como usuario se podrá seleccionar la posición en la que se colocará cada uno de los barcos siempre que cumpla con las restricciones del tablero y sin colocar un barco encima de otro. Se colocarán de forma aleatoria dentro del tablero siempre cumpliendo las restricciones del tablero y no podrá colocar un barco encima de otro.

Para un tablero de 10x10 se debe de utilizar los barcos de la siguiente forma:

- 1 Portaaviones
- 2 Submarinos
- 3 Destructores
- 4 Buques

Para un tablero de dimensiones $m \leq 20$ y $m > 10$ será el doble de barcos que un tablero de 10x10.

- 2 Portaaviones
- 4 Submarinos
- 6 Destructores
- 8 Buques

Para seleccionar el número de barcos para un tablero de dimensión m se aplicará la siguiente fórmula: $B(m) = \frac{(m-1)}{10} + 1$ (utilizando solo la parte entera, es decir sin aplicar los decimales) por el número de barcos en un tablero de 10x10.

Ejemplo para un tablero de dimensión m=24

$$B(m) = \frac{(24-1)}{10} + 1 = \frac{23}{10} + 1 = 2 + 1 = 3$$

Los barcos a utilizar para un tablero de dimensión m = 24 serán:

- 3 Portaaviones
- 6 Submarinos
- 9 Destructores
- 12 Buques

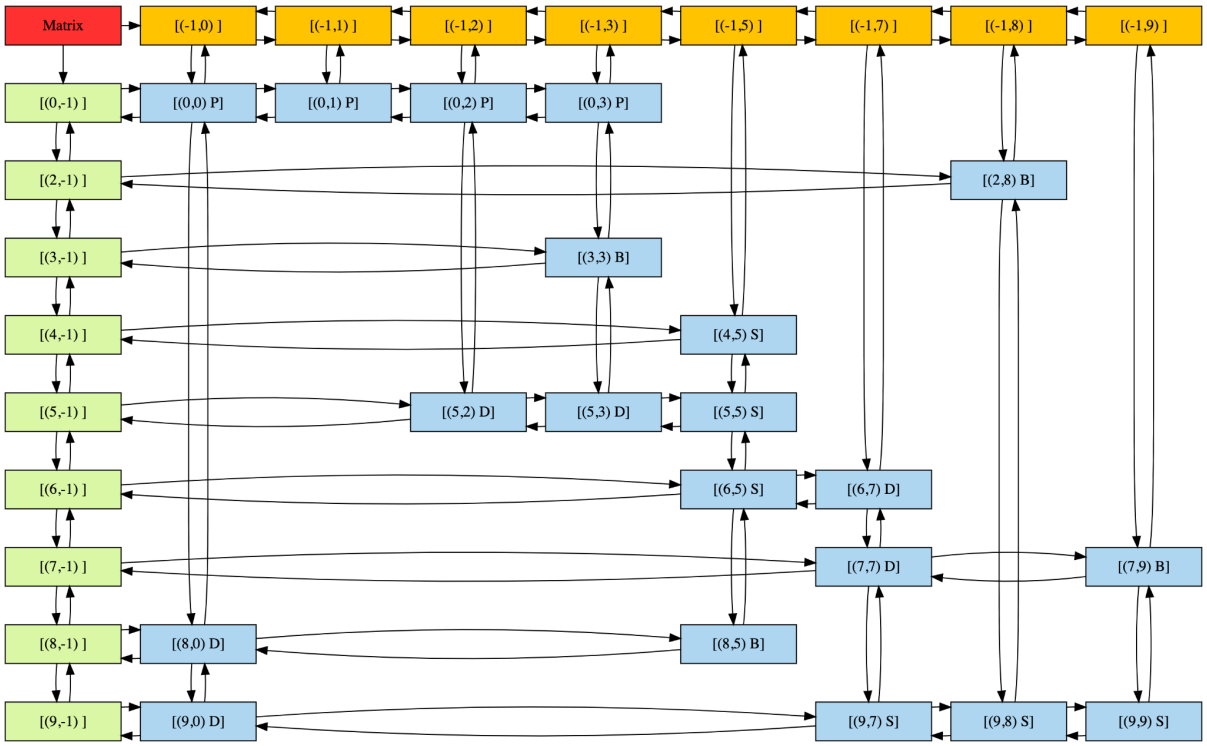
Se deben mostrar los dos tableros en juego. En el mapa principal se mostrarán los barcos ubicados inicialmente. Deben de mostrar el número correspondiente a cada fila y columna. Queda a discreción del estudiante la simbología a utilizar para un disparo y el límite del tablero.

El tablero debe ser construido utilizando como base una matriz dispersa, aunque el tablero tenga dimensiones definidas, únicamente se deben crear los nodos que contengan a las naves, y durante la realización de una partida se crearan nodos dinámicamente para los disparos ejecutados fuera de las posiciones ocupadas por las naves.

Ejemplo para un tablero de 10X10 a nivel lógico

- 1 Portaaviones = P
- 2 Submarinos = S
- 3 Destructores = D
- 4 Buques = B

Se utiliza la inicial de las naves para identificarlas dentro de la matriz, la forma de controlar el tipo de nave queda a discreción del estudiante.



Lógica del disparo

Al inicio del juego se debe de pedir las coordenadas (x, y) de cada disparo, y repintar el mapa por jugador hasta que sea el fin del juego.

- Cada disparo solo puede abarcar una casilla o una coordenada.
- Un barco quedará destruido cuando acierten los disparos en cada una de las posiciones de él dentro del tablero.
- Si el disparo acierta deberá notificar al jugador en curso
- Si las coordenadas no existen dentro del tablero en juego se debe de notificar al jugador en curso.

Descripción de las funcionalidades del juego solicitado

- Construir el juego Battleship utilizando interfaz gráfica construida utilizando python
- Modalidades del juego
    - Jugador 1

Lo primero que debe de mostrar el programa es un Menú de opciones.

Menú de ejecucion de una partida

1. Jugador solitario
    - Comenzará una partida modalidad un jugador
    - Se pedirá al jugador el ingreso del tamaño tablero a construir

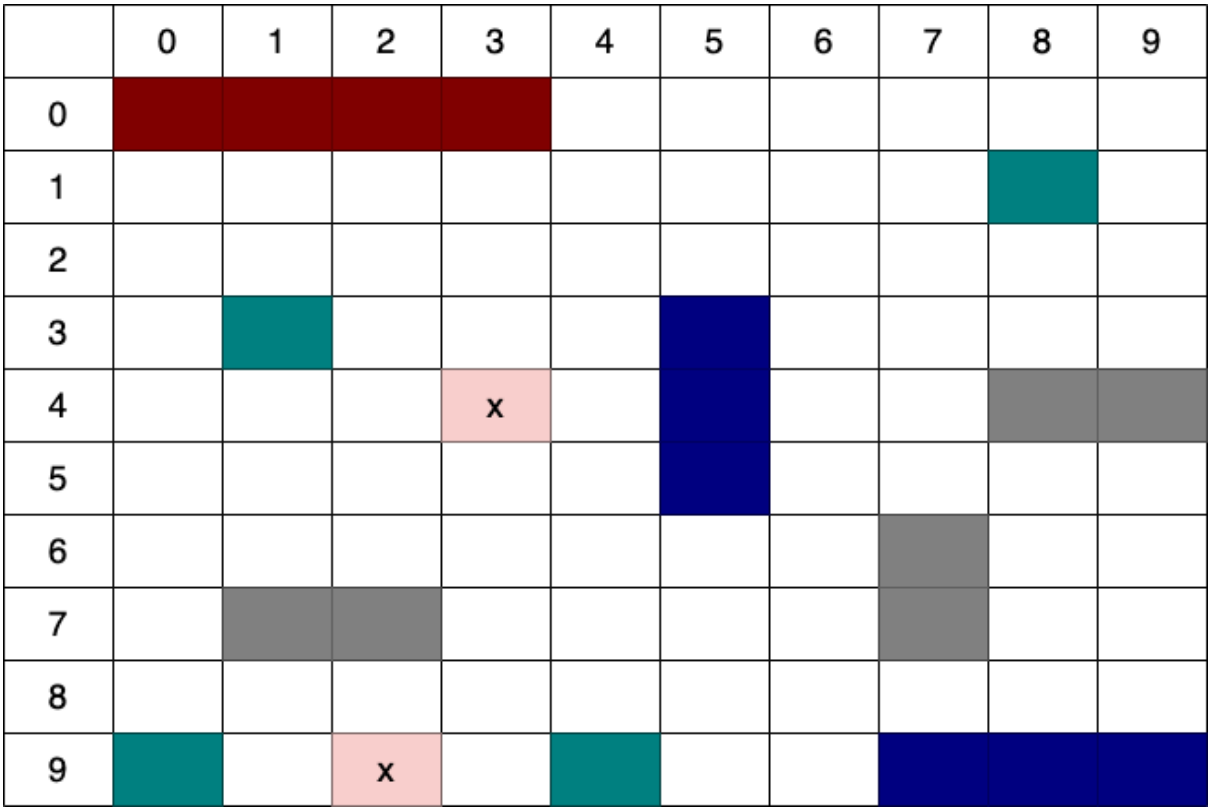
Partida

Para identificar las naves (sin skin) se utilizaran los siguientes colores:

Nave	Dimensiones (variable)	Color
Portaaviones	4x1	Maroon
Submarino	3x1	Navy
Destructor	2x1	Gray
Buque	1x1	Teal

Vista de un tablero de 10 x 10

3 vidas —-- 95 tokens disponibles



Durante la ejecución de una partida

1. Mostrar el estado actual del tablero
    a. Se debe mostrar el tablero principal en forma matricial.
2. Ejecutar movimiento del jugador
    a. En el caso de que el movimiento acierte con un barco, el jugador ganará 20 tokens (monedas del juego)
    b. En caso de que el movimiento sea inválido se descarta 1 vida del jugador.(el jugador únicamente cuenta con 3 vidas por partida).
3. Repetir los pasos anteriores hasta que no existan barcos enemigos o que las vidas del jugador se acaben.
4. El usuario podrá retroceder una movimiento pero perderá 5 tokens (monedas del juego)
    a. Eliminar del tablero el ataque
    b. Eliminar de la pila el movimiento
5. El usuario puede abandonar la partida y perderá 20 tokens (monedas del juego)

6. No se podrá contar con tokens negativos, por ende tendrá un límite la cantidad de retrocesos.

Al finalizar la partida

1. Mostrar los resultados del jugador, es decir el número de barcos destruidos y el número de errores.
2. Aumentar el registro de partidas del jugador.
    a. Registrar la partida en la lista de jugadas (FASE 1)
3. Preguntar si se desea comenzar una partida con la configuración ya establecida o si se desea regresar al menú principal.

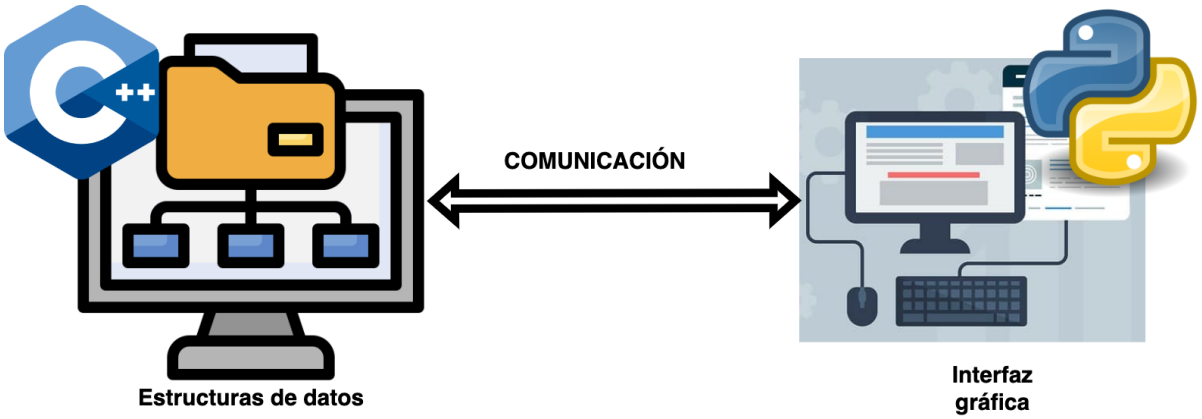
Funcionalidad de retroceso de jugadas para la modalidad de un jugador

Esta funcionalidad se describe en la fase 1 del proyecto, para esta fase se deberá mostrar de forma gráfica el estado del tablero en cada retroceso de jugada.

Tutorial del juego

Se debe implementar la visualización del tutorial de juego descrito en la fase 1, mostrando de forma gráfica la ejecución de cada paso del tutorial.

Arquitectura del juego



- **Estructuras y manejo de los datos:**

Se deben agregar a las estructuras de datos de la fase 1, las estructuras solicitadas en esta fase y proveerse del funcionamiento necesario para manejar todos los datos del juego: datos de usuarios, registro, inicio de sesión, artículos, compras, desarrollo del juego y todos los procesos necesarios para el correcto desempeño de la aplicación solicitada.

Esta sección de la aplicación debe codificarse en su totalidad utilizando el lenguaje de programación C++ a excepción de la matriz dispersa que se utilizará para la ejecución de una partida, es decir **la matriz dispersa es la única estructura de datos que deberá programar en Python**

- **Interfaz gráfica del juego.**

Este componente de la aplicación corresponde a las pantallas o vistas gráficas con las que el usuario va a interactuar, no se tendrá ninguna interacción por medio de la terminal o consola.

La interfaz gráfica debe desarrollarse utilizando el lenguaje de programación Python

- **Comunicación entre componentes**

Se debe proveer al juego de un mecanismo de comunicación entre la interfaz gráfica y el componente encargado de las estructuras de datos y manejo de la información, de tal forma que la interfaz gráfica solo se encargue de consumir y generar los datos que se almacenarán en el servidor de datos (c++).

Sección de estadísticas y visualizaciones gráficas

Visualizaciones gráficas

- Árbol de compras
- Matriz dispersa del tablero del tutorial
- Árbol B

Administración

Adicionalmente se debe contar con un módulo básico de administración únicamente para visualizar el listado de usuarios ordenado de forma ascendente y descendente como se había solicitado en la fase 1(utilizando la edad), además debe ser posible visualizar la gráfica del árbol B de usuarios.
Este usuario estará definido con la siguiente información

nickname: EDD
password: edd123
edad: 50

Siendo este el único usuario administrador.

Observaciones

- Lenguaje de Programación:
    - C++ para el desarrollo de las estructuras
    - Python para la interfaz gráfica
- Sistema Operativo: Elección del estudiante.
- El IDE a utilizar queda a discreción del estudiante.
- Librería para graficar las estructuras: Graphviz
- Los archivos de entrada serán documentos en formato JSON (.json)
- El estudiante debe tener un repositorio privado en github con el nombre [EDD_2S]BatallaNaval_#carnet, se crearán 3 carpetas fase1, fase 2 y fase 3.
- Agregar a su tutor como colaborador al repositorio del proyecto.
    - Auxiliar 1: wltomv
    - Auxiliar 2: aexlopez@gmail.com
    - Auxiliar 3: willop
- Se entregará en UEDI un .zip con los entregables solicitados.
- Las copias tendrán nota de 0 puntos y serán reportadas al catedrático y a la escuela de sistemas.

Entregables

- Manual Técnico
- Manual de usuario
- Compilado de C++
- Código fuente en un .rar
- Link a repositorio con el código fuente.

Restricciones

- Las estructuras deben de ser desarrolladas por los estudiantes sin el uso de ninguna librería o estructura predefinida en el lenguaje a utilizar.
- No se permite la modificación de código durante la calificación, únicamente se calificará sobre el ejecutable entregado en UEDI.

Fecha de Entrega

- 25 de septiembre a las 23:59 horas.
- **No se aceptarán entregas tarde**