Manual Técnico - Battleship Cliente-Servidor

Introducción

Este documento describe la arquitectura técnica, los diagramas de arquitectura, secuencia y clases, y las clases principales del proyecto **Battleship**, un juego de Batalla Naval cliente-servidor implementado en C++.

Arquitectura del Sistema

El sistema utiliza una arquitectura cliente-servidor con comunicación TCP. Los clientes (jugadores) se conectan a un servidor central que gestiona autenticación, emparejamiento y partidas.

Diagrama de Arquitectura

El siguiente diagrama muestra la estructura del sistema, incluyendo los componentes del cliente y servidor, y su interacción a través de TCP.

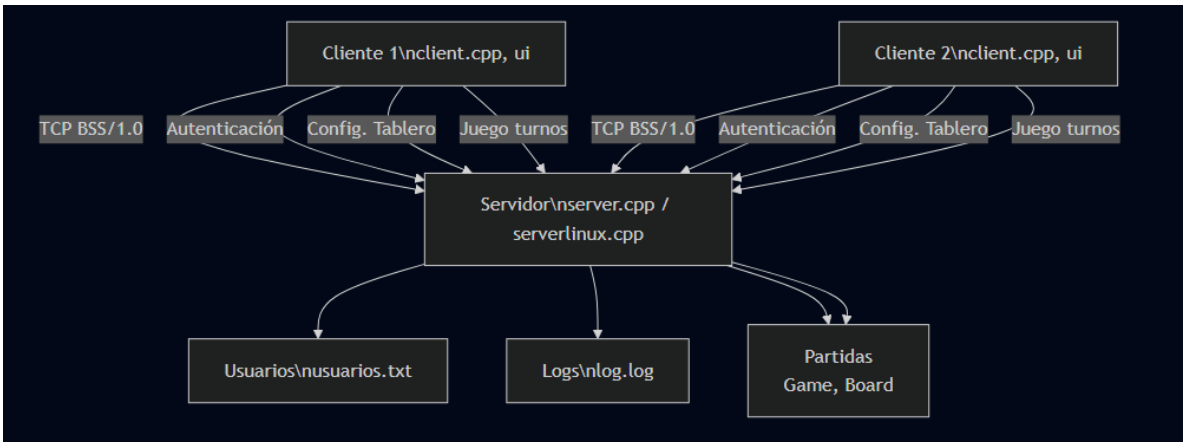


Diagrama de Secuencia.

El diagrama de secuencia muestra el flujo de una partida, desde el inicio de sesión hasta un disparo. Cubre la interacción entre dos clientes y el servidor.

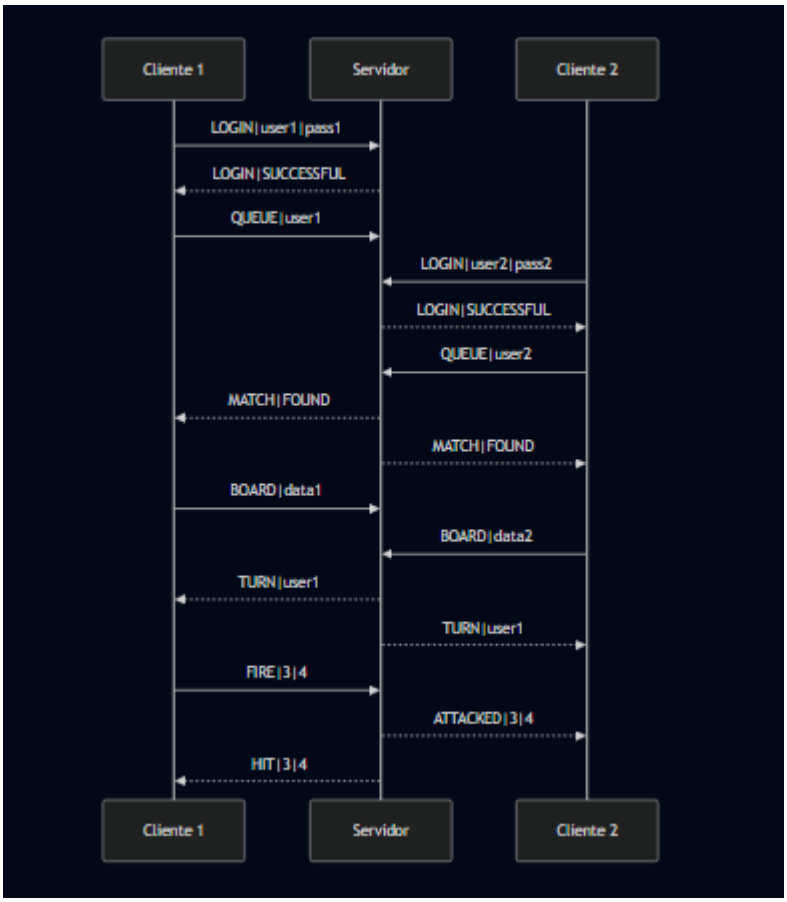
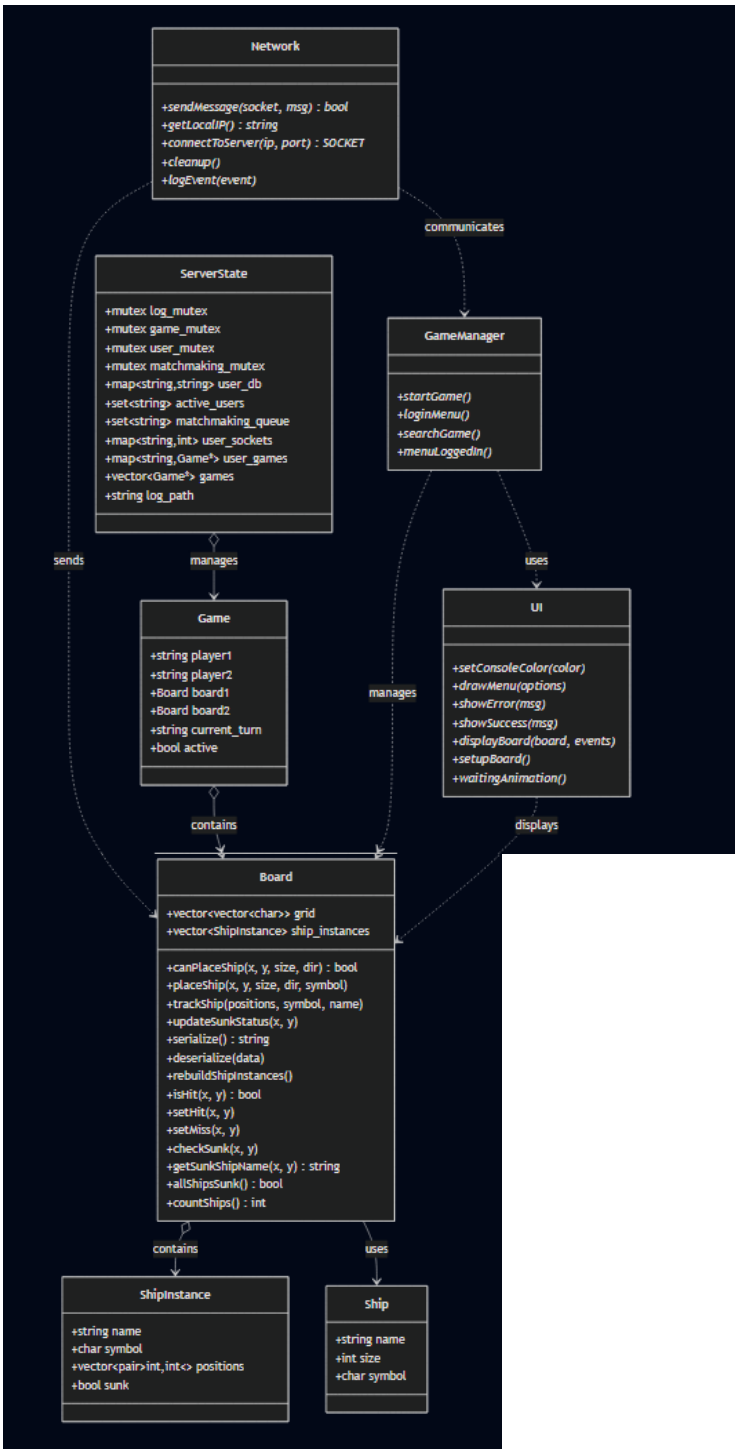


Diagrama de Clases.

El diagrama de clases detalla las principales clases y structs del proyecto, sus atributos, métodos y relaciones. Todas las clases están conectadas para reflejar sus interacciones en el sistema.



Clases Principales.

1. game_common.h

Ship (struct):

Atributos: name (nombre), size (tamaño), symbol (símbolo).

Uso: define tipos de barcos (ej. Portaaviones, Destructor).

Board (class):

Atributos:

- grid: matriz 10x10 para el tablero.
- ship_instances: lista de barcos colocados.

Métodos:

- canPlaceShip: verifica si un barco puede colocarse.
- placeShip: coloca un barco en el tablero.
- trackShip: registra posiciones de un barco.
- updateSunkStatus: actualiza el estado de hundimiento.
- serialize: convierte el tablero a una cadena.

2. game_manager.h

GameManager (class):

Métodos estáticos:

- startGame: inicia una partida.
- loginMenu: maneja autenticación.
- searchGame: busca oponentes.
- menuLoggedIn: menú para usuarios autenticados.

3. network.h

Network (class):

Métodos estáticos:

- sendMessage: envía mensajes al servidor.
- getLocalIP: obtiene la IP local.
- connectToServer: establece conexión TCP.
- cleanup: libera recursos de red.

- logEvent: registra eventos en un archivo.

4. ui.h

UI (class):

Métodos estáticos:

- setConsoleColor: cambia el color del texto.
- drawMenu: muestra menús.
- show fromError/Success: muestra mensajes.
- displayBoard: muestra el tablero.
- setupBoard: configura barcos.
- waitingAnimation: animación de espera.

5. server.cpp / serverlinux.cpp

- **Board** (class): similar a game_common.h, pero incluye deserialize y rebuildShipInstances.
- **Game** (struct): almacena jugadores, tableros y turno actual.
- **ServerState** (struct): gestiona estado global (usuarios, partidas, sockets).

Protocolo de Comunicación

Mensajes en formato COMANDO|PARAM1|PARAM2:

- REGISTER|user|pass: registrar usuario.
- LOGIN|user|pass: iniciar sesión.
- QUEUE|user: entrar en cola.
- FIRE|x|y: disparar.
- BOARD|data: enviar tablero.
- Respuestas: SUCCESSFUL, ERROR, HIT, MISS, SUNK, etc.

Consideraciones Técnicas

- **Multihilo**: el servidor usa hilos para emparejamiento y manejo de clientes.
- **Sincronización**: Mutex para acceso concurrente a datos (user_mutex, game_mutex).