El objetivo del informe es aclarar determinadas cosas que no están tan claras en el código o el diagrama.

## Comentarios

Todas las clases están comentadas, también sus atributos y métodos. Ante cualquier duda se puede acceder y leer los comentarios. Para los comentarios de clases y métodos, utilizamos

/// Sumary

También hay comentarios en algunos bloques de código, que ayudan a entender la lógica

## User – Player

User representa al usuario de Telegram en el programa.

Player es un atributo de User. La clase Player se encarga de conocer toda la información necesaria para el juego y tener todos los métodos necesarios para jugar. En un principio, Player podría no existir, y User tener los atributos y métodos de Player, pero eso no respetaría el SRP, por lo que decidimos separarlo en dos clases separadas, y se accede a Player a través de User.

## Board

El tamaño del tablero es de 10x10, predeterminado y fijo.

La cantidad máxima de barcos colocados es igual a 4.

## Excepciones:

El programa posee excepciones simples en varias zonas, para evitar cualquier tipo de error que pueda suceder.

También posee excepciones propias, las mismas son utilizadas por Handlers, Board, Logic y UserRegister. Abarcan casos en los que:

* Se ingresa mal el formato de la coordenada.
* Se ingresa mal una orientación de un barco.
* Se intenta colocar un barco cuando el tablero ya tiene los 4 barcos.
* Se intenta atacar por segunda vez en una ubicación.
* Se intenta utilizar un comando de handlers sin haber creado un usuario previamente.

La excepción UserNotCreatedException es utilizada por los handlers, sin embargo no está espefícado en el diagrama.

## Persistencia de datos

En nuestro caso, lo único que guardamos para que “sobreviva” es el número de id del último Game creado y el resumen de todos los juegos jugados hasta ahora.

Como en ninguno de los dos casos guardamos objetos, sino strings, decidimos aplicar una persistencia simple en un archivo txt.

En el caso del número de id, lo leemos del txt, lo utilizamos, lo incrementamos en 1, y lo volvemos a guardar. Para este caso sobrescribimos el archivo txt.

Para el resumen de juego, cada vez que un juego finaliza, se guarda el mismo en un archivo txt como una string, para eso se utiliza un método llamado GameToString, donde se crea un resumen con los datos más importante en forma de string. Cuando un usuario quiera ver todos los juegos que se han terminado de jugar hasta ahora, deberá ingresar el comando “ver partidas jugadas”, y se imprimirá frente a el todos los juegos terminados hasta ahora, de todos los user, organizados por la id del juego.

## Handler:

Los handler utilizan varias clases, sin embargo, no especificamos las relaciones en el diagrama porque si no se vería muy confuso, con flechas por todos lados

Para controlar las palabras clave que ningún Handler maneja, se decidió crea un FinalHandler, que se encarga de aceptar cualquier palabra y responder que no se entendió el comando ingresado.

## TelegramPrint: IIprinter

TelegramPrint es una implementación de IPrinter.

Existía el problema de que los handlers imprimían mensajes solamente en la pantalla de quienes escribían un comando, pero hay momentos, que es necesario que se imprima un mensaje específico en la pantalla del otro usuario.

Para ello se creó está clase. A través de la utilización del método SendTextMessageAsync del bot de Telegram(ITelegramBotClient) y la id del otro usuario, se imprime en la pantalla del usuario indicado (por su id) un mensaje correspondiente.

## Telegram

En Telegram, en cualquier momento puedes escribir la palabra “comandos” para observar en pantalla cuales son los comandos disponibles.

1. Primero que nada, debes crearte un usuario.
2. Luego puedes buscar partida o ver los resúmenes de todos los juegos jugados hasta ahora.

Puede elegir entre dos tipos de juego, normal y predictivo.

Mientras te encuentras en el Lobby (sala de espera), en cualquier momento puedes salir del mismo.

1. Una vez tengas pareja para jugar, se creará un juego con ambos y se te dirá la id del juego en la pantalla. Ahora debes colocar todos los barcos. La colocación en cuanto a los tamaños de los barcos irá en decreciente, primero colocaras el de 5 casillas y por último el de 2. En total debes colocar 4 barcos. Durante la colocación de barcos puedes observar el tablero.
2. Una vez que los dos jugadores coloquen todos los barcos. La partida se da a inicio, es un juego por turnos en el cual en el cual puede ver los tableros en cualquier momento. Las posibles acciones van desde un ataque normal, hasta las 3 posibles habilidades especiales, las cuales solo puedes utilizar una vez cada una. El juego finaliza cuando uno de los dos jugadores hunda todos los barcos del contrincante. Una vez terminado, el juego se guardará y volverás al paso 2.

## Funciones extras

1. Modo de juego con disparos predictivos. Cada vez que un jugador ataque y su resultado sea “agua”, se analizaran las 4 casillas alrededor de la coordenada de ataque, y en caso de que una de las cuatro casillas contenga un barco, en lugar de decir “agua” dirá “agua casi tocado”.

Lo que cambia con el modo de juego normal, es el procedimiento que sucede durante el ataque, por lo que para implementarlo basto con crear un método nuevo dentro de la lógica, similar al método Attack, llamado AttackPredictive, que agregue las nuevas funciones.

1. Cada jugador, en cualquier momento de la partida, puede realizar un ataque aéreo. El ataque aéreo consiste en atacar a toda una fila con un solo turno. Solo se puede usar una vez por partida.
2. El tablero se divide en 2 sectores (superior e inferior). Con esta habilidad el jugador puede obtener información sobre en cuál de los sectores existen más puntos de impacto. Solo se puede usar una vez por partida.
3. Cualquier usuario puede acceder a los resúmenes de todos los juegos jugados por todos los users hasta ahora.

La implementación esta explicada en la sección de persistencia de datos.

1. Cada jugador, en cualquier momento de la partida, puede ingresar una columna del tablero del contrincante, y obtendrá una “foto satelital” de la misma.

Para la implementación del 1,2 y 4, se crearon nuevo Handlers, nuevos métodos en Logic, y se sumaron como una habilidades especiales dentro de Player. También se creó la clase SpecialHabilities, encapsulándola y conteniéndola dentro del Player, de forma que si en el futuro se quieren agregar más, entonces se modifica SpecialHabilities en lugar de Player