Identificación del problema y análisis de requerimientos

Caso de Estudio :Juego Batalla Naval

Cliente	**Facultad de Negocios y Economía**
Usuario	Estudiantes de economía y negocios internacionales, y todo aquel que juegue para poner a prueba sus comportamientos
Requerimientos funcionales	*[RF1:* Permitir al jugador humano ubicar sus barcos *[RF2: Ejecutar una jugada por turno verificando que la coordenada sea válida]* *[RF3: Determinar cuando y quien gana el juego]* *[RF4: Permitir varias iteraciones del juego, llevando un recuento de victorias de cada parte, que será mostrado cada final de partida o cada que el usuario acceda desde el menú principal]*
Contexto del problema	*Desde la facultad de negocios y economía, se están realizando estudios acerca de la teoría de juegos. Por tal razón están solicitando el diseno y creación de un software que simule el juego "Batalla Naval". En esta primera fase hay que tener en cuenta: Son 2 jugadores: El usuario, quien estará escribiendo coordenadas válidas y la máquina, que dará valores aleatorios. Además, se debe permitir varias iteraciones del juego, llevando un recuento de victorias. Por último, cabe destacar que el juego debe ser diseñado bajo las normas del juego ya establecidas en el enunciado.*
Requerimientos no funcionales	RNF1. Desarrollar el juego en una dimensión en Java. RNF2. La interfaz del usuario del juego debe ser clara y comprensible RNF3. El programa debe ejecutarse en menos de 1 segundo por turno RNF4. Experiencia de usuario fluida.
Requerimientos de proceso	El desarrollo del juego debe estar listo para la segunda semana del mes de abril. (7/04/2025)

.

Identificador y nombre	*[RF1: Permitir al jugador humano ubicar sus barcos]*

Resumen	*Permite al usuario turno a turno ubicar sus barcos en orden, en d*		
Entradas	**Nombre entrada**	**Tipo de dato**	**Condición valores válidos**
	posicionBarco	int	*Número entero entre 1 1 y 10, Que no se encuentren barcos previamente ubicados en esa posición*
Resultado o Postcondición	Se valida que no exista ningún barco en esa posición y se guarda en el arreglo un valor id		
Salidas	**Nombre salida**	**Tipo de dato**	**Formato**
	Barcos	Array	Se muestra el arreglo de barcos actualizado

Identificador y nombre	*RF2: Ejecutar una jugada por turno verificando que la coordenada sea valida.*		
Resumen	*Cuando el turno sea del jugador humano, se le debe permitir escribir una coordenada para atacar un barco enemigo. El juego de la máquina se hace de manera aleatoria. Después de cada turno, se modifica y muestra el estado actual de cada línea de mar.*		
Entradas	**Nombre entrada**	**Tipo de dato**	**Condición valores válidos**
	ataque	int	*Número entero entre 1 1 y 10. Que no haya sido digitado en anteriores turnos.*

Resultado o Postcondición	Teniendo en cuenta los barcos ubicados aleatoriamente por la máquina, se le dice al usuario si en la coordenada atacada tocó o no a un barco enemigo.		
Salidas	**Nombre salida**	**Tipo de dato**	**Formato**
	tocado	String	“Le diste a un barco”
	noTocado	String	“Fallaste”
	tableroCPU	Array	Array de 10 números enteros teniendo en cuenta: 0 (Agua), 2 (Parte tocada de barco enemigo, 3 (Barco totalmente hundido)
	tableroJugador	Array	Array de 10 números enteros teniendo en cuenta: 0 (Agua), 1(Barco perteneciente al humano), 2 (Parte tocada de barco enemigo), 3 (Barco totalmente hundido)

Identificador y nombre	*RF3: Determinar cuándo y quién gana el juego.*		
Resumen	*Cuando un jugador haya derrumbado todos los barcos enemigos, el juego termina y se debe decir quién ha sido el ganador.*		
Entradas	**Nombre entrada**	**Tipo de dato**	**Condición valores válidos**
Resultado o Postcondición	Cuando en la línea de mar de un jugador solo aparecen 0 (Agua) y 3 (barco totalmente hundido) a este se le considera perdedor, y al otro ganador.		
Salidas	**Nombre salida**	**Tipo de dato**	**Formato**
	Ganador	String	*“El jugador ganador ha sido [Humano o máquina]”*

Identificador y nombre	*[RF4: Permitir varias iteraciones del juego]*		
Resumen	*El juego debe permitir varias iteraciones llevando un recuento de victorias de cada parte, que será mostrado cada final de partida o cada que el usuario acceda desde el menú principal*		
Entradas	**Nombre entrada**	**Tipo de dato**	**Condición valores válidos**
	victoria	int	*número entero valor 1 que se suma al contador de victorias de cada una de las partes dependiendo de quién haya ganado*
	validacion	int	Se le pregunta al usuario si desea seguir jugando, si marca 1 se empieza una nueva partida, si marca 0, se cierra el programa
Resultado o Postcondición	Se suma el valor de la última victoria a quien corresponda, además se muestra el valor de victorias totales. Se pregunta si se desea seguir jugando.		
Salidas	**Nombre salida**	**Tipo de dato**	**Formato**
	victoriasTotalesCPU	int	*sumatoria de victorias totales de la máquina*
	victoriasTotalesUsr	int	sumatoria de victorias totales del usuario
	validacionError	String	Ha habido un error, asegúrese de que el dato ingresado sea un número entre 0 y 1