

### **Tarea 3: Documentación**

#### **Paradigma Imperativo y Orientado a Objetos: breakOutTec**

José Julián Camacho Hernández

Jose Fabián Mendoza Mata

Sergio Andrés Ríos Campos

Área de Ingeniería en Computadores, Instituto Tecnológico de Costa Rica

CE-3104: Lenguajes, Compiladores e Intérpretes

Profesor: Marco Rivera Meneses

30 de octubre de 2020

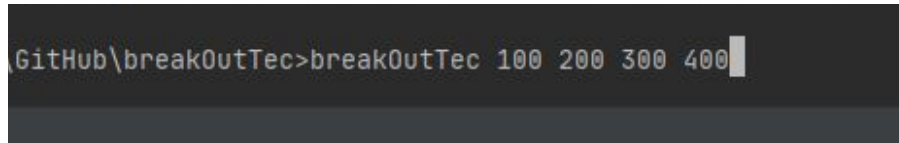
## **Tabla de contenidos**

Breve descripción del proyecto	3
1.1. Manual de usuario: cómo ejecutar el programa	3
1.2. Descripción de la utilización de las estructuras de datos desarrolladas.	6
1.3. Descripción detallada de los algoritmos desarrollados	8
1.4. Problemas sin solución.	9
1.5. Plan de Actividades realizadas por estudiante.	9
1.6. Problemas encontrados.	11
1.7. Conclusiones del proyecto	11
1.8. Recomendaciones del proyecto.	12
1.9. Bibliografía consultada en todo el proyecto	13
2. Bitácora en digital, donde se describen las actividades realizadas.	14

## 1. Breve descripción del proyecto

BreakOutTec es un juego inspirado en el videojuego arcade Breakout. Consiste en una superficie en la parte inferior de la pantalla que simula una raqueta que el jugador puede desplazar de izquierda a derecha. En la parte superior se sitúa un conjunto de ladrillos. El jugador debe golpear una pelota con la raqueta, haciendo desaparecer los ladrillos tocados por la pelota. El objetivo del juego es terminar con la pared de ladrillos.

### 1.1. Manual de usuario: cómo ejecutar el programa

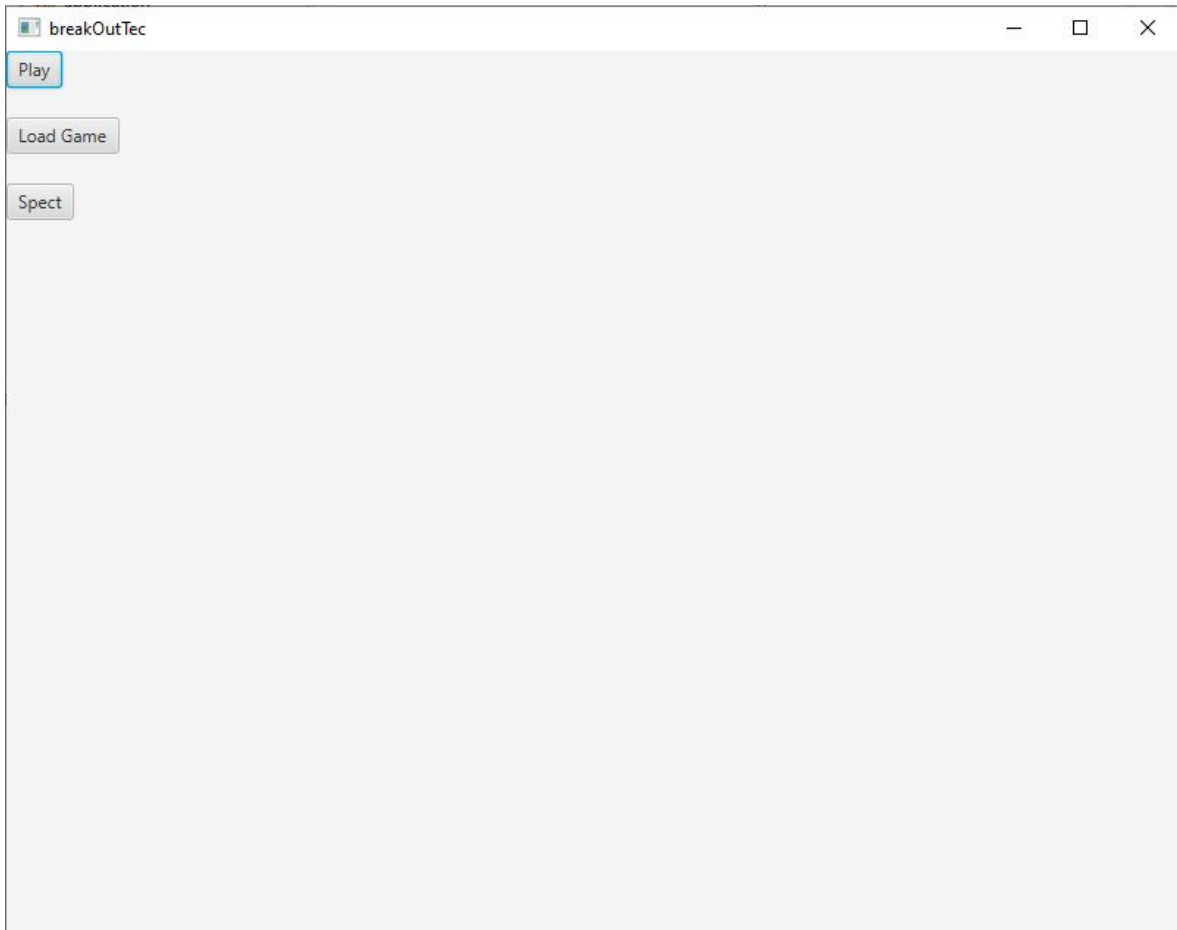
A screenshot of a terminal window with a dark background. The prompt is 'GitHub\breakOutTec>' and the command entered is 'breakOutTec 100 200 300 400'. A white cursor is at the end of the command.

```
GitHub\breakOutTec>breakOutTec 100 200 300 400
```

Para ejecutar el servidor, se debe escribir el siguiente comando estando en la ubicación de breakOutTec. Es importante que este se ejecute antes del cliente.

Para ejecutar el cliente debe ejecutarse la función main de la clase Main preferiblemente desde el IDE de Eclipse.

El programa se presenta al usuario en el menú principal del juego, se muestra una interfaz que contiene 3 botones:



- Jugar: Cuando se presiona, el juego establece una conexión con el servidor de manera que éste le pasa un JSON con las constantes, variables y matriz del juego.

Luego, se presenta una nueva ventana, que contiene el juego, presenta la matriz, las bolas con las que se jugarán el nivel y el jugador, el cual es la raqueta.

Movilidad: El jugador se puede mover a la derecha con la tecla A y a la izquierda con la tecla D por en la pantalla.

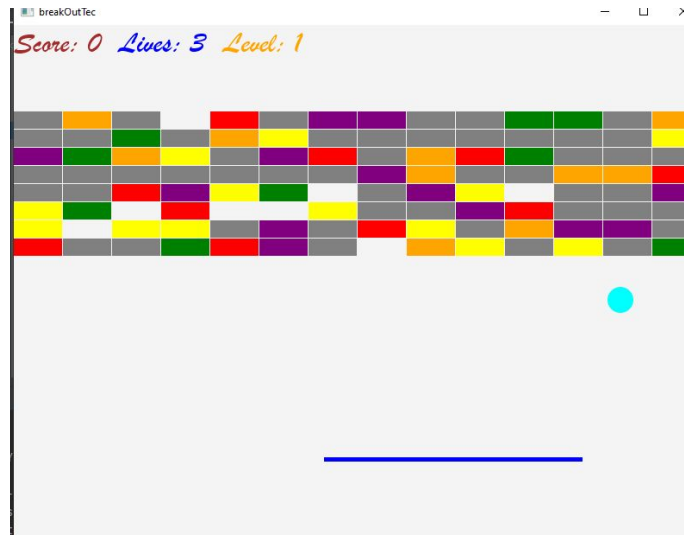
La pelota rebota tanto en el jugador como en los ladrillos, el objetivo es que la bola no pase por debajo de la raqueta, además de conseguir romper todos los ladrillos para pasar de nivel.

Cada bloque puede afectar al juego en:

- Cantidad de vidas del jugador.
- Cantidad de bolas en juego.
- Largo de la raqueta

- Velocidad de las bolas en juego

Estos bloques especiales aparecerán de diferente color en la interfaz, indicando que cambian los elementos del juego.



**Figura 1.** Funcionamiento del juego.

Ladrillo Normal: Gris

Ladrillo Vida: Verde

Ladrillo Bola: Amarillo

Ladrillo Incrementar Velocidad: Naranja

Ladrillo Disminuir Velocidad: Morado

Ladrillo Aumentar Raqueta: Rojo

Ladrillo Disminuir Raqueta: Negro

- Load Game: Obtiene el juego en formato JSON del servidor, lo carga en memoria y, al presionar Jugar, carga las variables en la interfaz.
- Spect: Ventana donde se puede espectar una partida que se está jugando.

## 1.2. Descripción de la utilización de las estructuras de datos desarrolladas.

### Estructura del juego

La estructura del juego consta de una serie de variables que son miembros de un struct en C. Dentro de ellas, se encuentra un arreglo bidimensional que representa el conjunto de ladrillos en el juego. Tiene dimensiones 8 X 14 predeterminadamente, y puede contener números enteros del cero al diez dependiendo de lo que representa cada ladrillo de la siguiente manera:

- 0: No existe ladrillo en ese espacio.
- 1: En esa ubicación se encuentra un bloque color verde.
- 2: En esa ubicación se encuentra un bloque color amarillo.
- 3: En esa ubicación se encuentra un bloque color naranja.
- 4: En esa ubicación se encuentra un bloque color rojo.
- 5: En esa ubicación se encuentra un ladrillo de aumento de vida.
- 6: En esa ubicación se encuentra un ladrillo de aumento de la cantidad de bolas.
- 7: En esa ubicación se encuentra un bloque de aumento de velocidad.
- 8: En esa ubicación se encuentra un ladrillo de decremento de velocidad.
- 9: En esa ubicación se encuentra un ladrillo de aumento del tamaño de la raqueta.
- 10: En esa ubicación se encuentra un bloque de decremento del tamaño de la

raqueta.

```
matrix :
[
  [4,4,4,4,4,4,4,4,4,4,4,4,4,4],
  [4,4,4,4,4,4,4,4,4,4,4,4,4,4],
  [3,3,3,3,3,3,3,3,3,3,3,3,3,3],
  [3,3,3,3,3,3,3,3,3,3,3,3,3,3],
  [2,2,2,2,2,2,2,2,2,2,2,2,2,2],
  [2,2,2,2,2,2,2,2,2,2,2,2,2,2],
  [1,1,1,1,1,1,1,1,1,1,1,1,1,1],
  [1,1,1,1,1,1,1,1,1,1,1,1,1,1]
],
```

**Figura 2.** Matriz inicial del juego

La estructura principal del juego también cuenta con miembros como el número de vidas, su puntaje, cantidad de bolas, velocidad de las bolas, tamaño de la raqueta, y sus ubicaciones.

## Diagrama de clases y diagrama de arquitectura

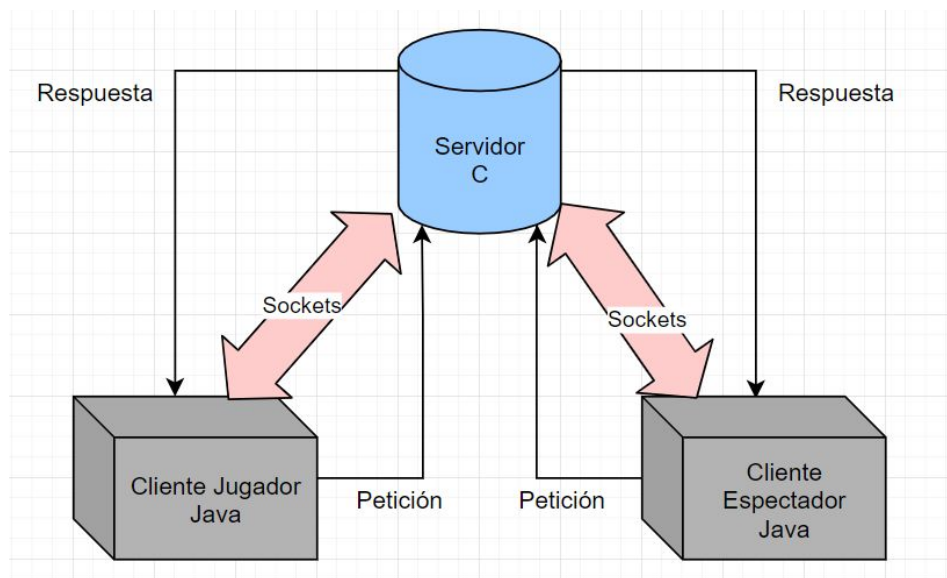


Figura 3. Diagrama de arquitectura

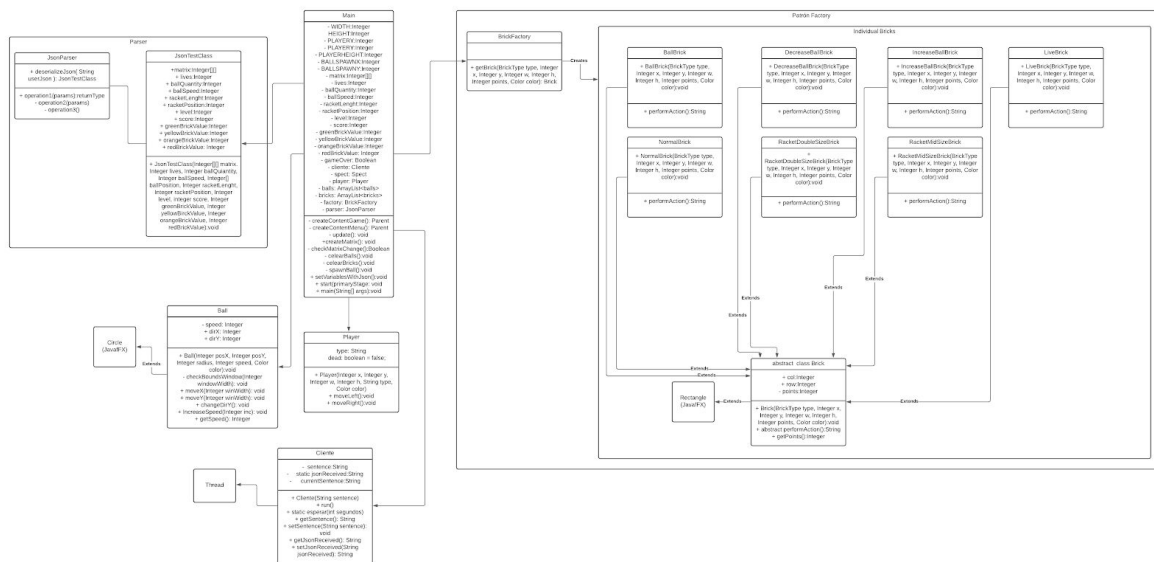


Figura 4. Diagrama de clases (Cliente)



### 1.3. Descripción detallada de los algoritmos desarrollados

#### Servidor

**receiveClientMessage(char\* msg):** es un algoritmo para la interpretación del mensaje recibido en el servidor desde el cliente para así modificar la estructura del juego. Se encarga de dividir el mensaje de la siguiente manera: la primera palabra corresponde con el evento que sucedió en el cliente, por ejemplo “Broke”. Lo que le sigue son las coordenadas o índices en la matriz del ladrillo que se quebró. Por lo tanto, un mensaje válido sería por ejemplo “Broke 7 7”. Si el mensaje no es válido lo omite.

Una vez el mensaje fue analizado, se procede a la modificación de la estructura del juego. De esa manera, si un ladrillo se quebró, se realizan las siguientes acciones:

1. Se busca en la matriz según sus coordenadas.
2. Posteriormente, se pone el valor de ese ladrillo en cero para indicar que ya fue roto.
3. Se analiza el número que indica el tipo de ladrillo y se realiza la acción correspondiente, ya sea simplemente aumentar el puntaje o dar una vida extra, una bola extra, entre otras.

**receiveUserMessage(char\* msg):** corresponde a un algoritmo que analiza el mensaje del usuario desde consola para así modificar la estructura del juego. Este funciona de manera similar al anterior. De igual manera obtiene y divide el mensaje. Posteriormente, analiza las coordenadas dependiendo del valor de estas, se coloca el número identificador del ladrillo que el usuario desea poner en ese lugar. De ese modo el cliente recibe la matriz modificada y la muestra en pantalla.

**jsonGame(char\* p):** se encarga de convertir la estructura actual del juego en formato de una cadena tipo json. Para ello, utiliza funcionalidades de las librerías incluidas en el punto siete y ocho de la bibliografía, insertando en la cadena los valores de cada uno de los miembros de la estructura principal del juego.

#### **1.4. Problemas sin solución: En esta sección se detalla cualquier problema que no se ha podido solucionar en el trabajo.**

- Se presentó un problema que no pudo ser solucionado relacionado a la conexión mediante sockets del cliente y el servidor. Se presenta un problema a la hora de enviar mensajes de vuelta al cliente, ya que no se logra enviar de inmediato la matriz actualizada, sino que esta se envía de manera retrasada. Debido a ello, en el cliente no se puede mostrar en cada momento el juego actual, aunque en el servidor sí se encuentra la matriz del juego actual.
  
- No se logró la implementación del cliente espectador.

#### **1.5. Plan de Actividades realizadas por estudiante.**

<b>Tarea</b>	<b>Tiempo Estimado</b>	<b>Responsable</b>	<b>Fecha de entrega</b>
Crear la estructura del juego según los requerimientos	2 horas	Julián Camacho	21/10/2020
Preparar el socket del server en C	1 hora	Sergio Ríos	21/10/2020

Preparar el socket del cliente en java	1 hora	Sergio Ríos	22/10/2020
Crear interfaz donde se muestre el juego	4 horas	Fabián Mendoza	22/10/2020
Implementación de funciones para la modificación estructura del juego según parámetros de entrada y según los mensajes esperados	8 horas	Julián Camacho	23/10/2020
Lectura de JSON en el cliente	3 horas	Fabián Mendoza	23/10/2020
Cambio de variables en el juego dependiendo del JSON	4 horas	Fabián Mendoza	23/10/2020
Implementación de los sockets a la lógica y a la interfaz del juego, resolviendo ciertos problemas de compatibilidad	5 horas	Sergio Ríos	25/10/2020
Convertir la estructura del juego a JSON para ser enviada por sockets	10 horas	Julián Camacho	25/10/2020

### **1.6. Problemas encontrados: descripción detallada, intentos de solución sin éxito, soluciones encontradas con su descripción detallada, recomendaciones, conclusiones y bibliografía consultada para este problema específico**

- i) En la conexión entre C (Server) y Java(Cliente), se notó un problema que a la entrega de este proyecto, no se obtuvo solución. El problema consiste en la lectura errónea por parte del server de los mensajes enviados por el cliente, ya que, el server lee y registra dos mensajes separados enviados por el cliente, el primer mensaje se trata del primer caracter del dato enviado por el cliente, mientras que el segundo mensaje consiste en el resto del dato enviado por el cliente. Ya que no se pudo saber la causa de este error, se procedió a manejarlo de la siguiente manera: Se insertó un caracter predeterminado al inicio de todos los mensajes enviados por el cliente, este caracter se trata de un slash (/), de esta manera el server recibe el primer caracter, que se trata del slash, mientras que el segundo caracter es el mensaje completo del cliente. De esta manera se puede descartar el primer caracter (/) y trabajar con el segundo, el cual representa el mensaje completo del cliente.

### **1.7. Conclusiones del proyecto**

- i) Se concluye que desarrollar una aplicación utilizando C y Java es una opción viable, debido a que existe manera de comunicar ambos lenguajes e integrar el funcionamiento entre los paradigmas de programación imperativo y orientado a objetos.
- ii) Se logró demostrar que los lenguajes C y Java pueden compartir información mediante sockets, facilitando el desarrollo de aplicaciones que involucren ambos lenguajes.
- iii) Se logró modelar el problema mediante la implementación de diversas estructuras y clases. Por lo anterior se concluye que para el desarrollo de un juego como BreakOut, la utilización de los paradigmas imperativo y orientado a objetos es conveniente.
- iv) Se logró implementar el juego mediante el manejo de listas, estructuras que pueden ser fácilmente utilizadas en ambos lenguajes.

## 1.8. Recomendaciones del proyecto.

- a. Para proyectos de este tipo, se recomienda la utilización del paradigma de programación orientado a objetos e imperativo. Esto debido a que el problema puede ser modelado y solucionado de manera natural utilizando ambos paradigmas.
- b. Se recomienda la utilización de sockets para la comunicación e intercambio de información entre Java y C, ya que es una manera óptima de conectar una estructura cliente - servidor entre ambos lenguajes.
- c. Al tomar la decisión de trabajar en el sistema operativo Windows, ciertos aspectos en la parte de conexión se tienen que tomar en cuenta, por ejemplo, el uso de la librería Winsock y el link que se debe de hacer con ws2\_32, esto se realiza mediante la bandera "-lws2\_32" que se debe colocar en el tiempo de compilación del servidor.
- d. Debido a ciertos aspectos de conexión se recomienda usar el juego en un ambiente de desarrollo Windows.

## 1.9. Bibliografía consultada en todo el proyecto

CreateThread function (processthreadsapi.h) - Win32 apps. (2018). Recuperado el 13 de Octubre 2020 de

<https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createthread>

C - Input and Output - Tutorialspoint. (2020). Recuperado el 13 de Octubre 2020 de

[https://www.tutorialspoint.com/cprogramming/c\\_input\\_output.htm](https://www.tutorialspoint.com/cprogramming/c_input_output.htm)

Command line arguments in C/C++ - GeeksforGeeks. (2017). Retrieved 26 October 2020,

from <https://www.geeksforgeeks.org/command-line-arguments-in-c-cpp/>

ASystemProgramming Channel. (2020). *13.Thread Synchronization Using CreateMutex & CreateThread API- WIndows System Programming in C/C++*.

YouTube. Recuperado el 13 de Octubre 2020 de <https://www.youtube.com/watch?v=VpXQYgeWvf4&t=302s>.

Thakur, Arjun. *How do I convert a char to an int in C and C++?*. (2020). Recuperado el 13 de Octubre 2020 de

<https://www.tutorialspoint.com/how-do-i-convert-a-char-to-an-int-in-c-and-cplusplus>

variables, C., Core, D., Wei, H., & Mohammad, K. (2012). *C split a char array into different variables*. Recuperado el 13 de Octubre 2020 de

<https://stackoverflow.com/questions/10349270/c-split-a-char-array-into-different-variables>

rafagafe/json-maker. (2020). Recuperado el 13 de Octubre 2020 de

<https://github.com/rafagafe/json-maker>

rafagafe/tiny-json. (2020). Recuperado el 13 de Octubre 2020 de

<https://github.com/rafagafe/tiny-json>

## 2. Bitácora en digital, donde se describen las actividades realizadas.

Fecha	Estudiante(s)	Actividad
15/10/2020	Todos	Primera reunión para comprender, organizar y dividir el trabajo correspondiente a cada uno de los integrantes de acuerdo con la dificultad estimada de este.
17/10/2020	Sergio Ríos	Comienzo de investigación acerca de conexiones entre C y Java mediante sockets
17/10/2020	Todos	Decisión del sistema operativo a utilizar en la construcción del proyecto
17/10/2020	Julián Camacho	Modelado de la estructura del juego en el servidor. Tercer punto de la bibliografía para reconocer los argumentos brindados por el usuario.
20/10/2020	Sergio Ríos	Investigación sobre el uso de la librería Winsock para la conexión del server
20/10/2020	Julián Camacho	Implementación de funciones para la modificación de la estructura del juego dependiendo del mensaje recibido. Investigación del uso de hilos para escuchar los mensajes del usuario desde consola (Punto uno dos, tres y cuatro de la bibliografía).
21/10/2020	Sergio Ríos	Construcción de sockets
22/10/2020	Julián Camacho	Continuación del desarrollo de funciones para la modificación de la estructura del juego.

22/10/2020	Fabián Mendoza	Creación de la interfaz para el juego
23/10/2020	Fabián Mendoza	Creación de un lector en formato JSON para java, comunicarlo con el cliente.
24/10/2020	Julián Camacho	Investigación de bibliotecas y funciones para generar un json a partir de la estructura del juego. Se utilizaron los puntos siete y ocho de la bibliografía.
24/10/2020	Fabián Mendoza	Cambiar variables del juego dependiendo de un archivo JSON.
25/10/2020	Sergio Ríos	Implementación de sockets con lógica e interfaz del juego
28/10/2020	Sergio Ríos Julian Camacho	Resolución de problemas con el socket por parte del server
30/10/2020	Julián Camacho Fabián Mendoza	Resolución de problemas de comunicación entre cliente y servidor en la interfaz del juego.