

JOSE ALBERTO MIÑON VELAZQUEZ
314202226

Informe y Documentación

Descripción del protocolo de comunicación diseñado

El protocolo de comunicación implementado para el juego de Tic-Tac-Toe se basa en el intercambio de mensajes en formato JSON entre el cliente y el servidor, utilizando sockets TCP. Los clientes envían solicitudes al servidor indicando las acciones que desean realizar (por ejemplo, registro, inicio de sesión, hacer un movimiento en el juego) y el servidor responde con el estado actualizado del juego o el resultado de la operación solicitada.

Cada mensaje sigue una estructura definida que incluye:

- *action*: Indica la operación a realizar, como register, login, makeMove, etc.
- *data*: Contiene la información relevante para la operación, como las credenciales del usuario, el símbolo del jugador o las coordenadas del movimiento en el tablero.

El servidor procesa las solicitudes y devuelve respuestas JSON que contienen:

- *status*: Indica el resultado de la operación (success o error).
- *data*: Proporciona información adicional, como el estado actualizado del tablero, el siguiente jugador o un mensaje de error.

Manejo de autenticación y persistencia de la sesión utilizando cookies

La autenticación se gestiona mediante un sistema de registro e inicio de sesión, en el cual el cliente proporciona un nombre de usuario y una contraseña. Tras un registro exitoso, el cliente puede iniciar sesión, y el servidor devuelve un identificador de sesión (**sessionId**) único para cada usuario autenticado.

Este identificador de sesión se guarda en el cliente y se envía junto con cada solicitud futura para identificar al usuario en el servidor. El uso de cookies para la persistencia de la sesión permite mantener al usuario autenticado durante el transcurso del juego y evitar la necesidad de repetir el inicio de sesión.

Manejo del estado de la partida y sincronización de los clientes con el servidor

El estado de la partida se almacena y actualiza en el servidor. Cada vez que un jugador realiza un movimiento, el servidor actualiza el estado del tablero y verifica si hay un ganador. El servidor se encarga de enviar el estado actualizado del juego a todos los clientes conectados, sincronizando así a todos los jugadores.

Para mantener la persistencia del estado del juego, el servidor guarda el estado del tablero y los turnos en un archivo JSON llamado "game_state.json". De esta manera, si el servidor

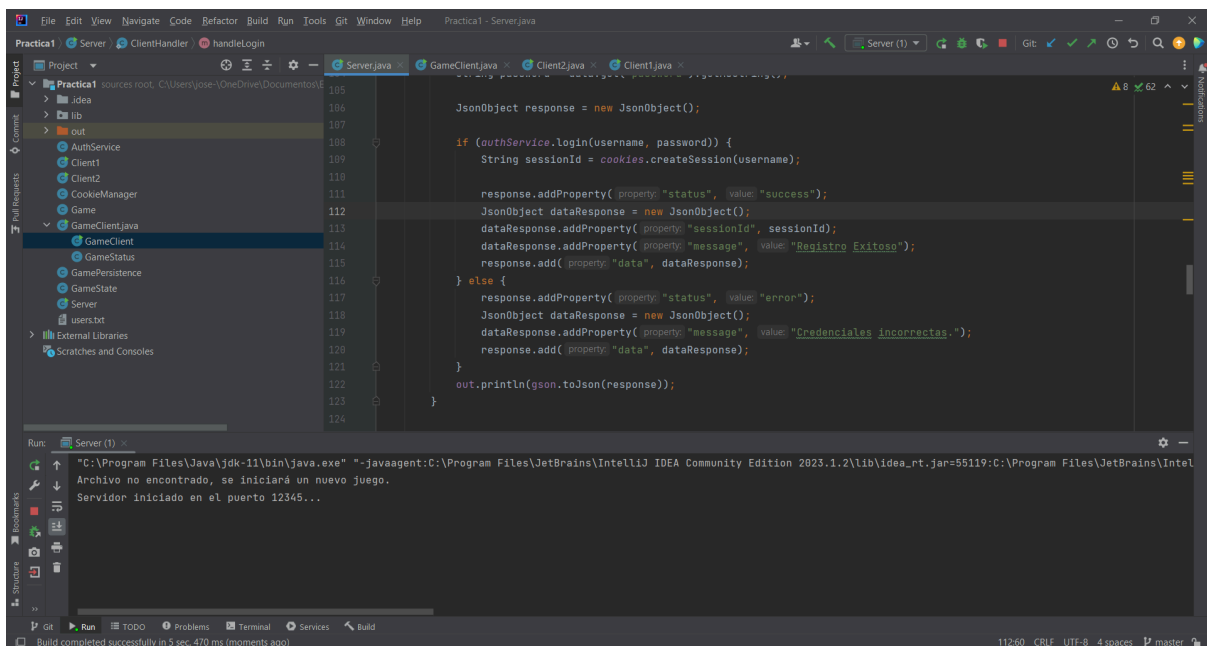
se reinicia o la partida se pausa, se puede cargar el estado previamente guardado desde el archivo y continuar la partida.

El proceso de sincronización incluye:

1. El cliente realiza un movimiento y lo envía al servidor.
2. El servidor valida el movimiento, actualiza el tablero y determina si hay un ganador.
3. El servidor envía el estado actualizado a todos los clientes excepto al que envió el movimiento, garantizando que todos vean la misma versión del tablero.
4. Los clientes muestran el tablero actualizado en sus interfaces.

PRUEBA

1.- Primero ejecutamos la clase server



```
File Edit View Navigate Code Refactor Build Run Tools Git Window Help Practica1 - Server.java
Practica1 Server ClientHandler handleLogin
Project
  Practica1
    sources root, C:\Users\jose-1\OneDrive\Documents\Idea
    idea
    lib
    out
    AuthService
    Client1
    Client2
    CookieManager
    Game
    GameClient.java
    GameClient
    GameStatus
    GamePersistence
    GameState
    Server
    users.txt
    External Libraries
    Scratches and Consoles
Run: Server (1)
"C:\Program Files\Java\jdk-11\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.1.2\lib\idea_rt.jar=55119:C:\Program Files\JetBrains\Intel
Archivo no encontrado, se iniciará un nuevo juego.
Servidor iniciado en el puerto 12345...
Build completed successfully in 5 sec 470 ms (moments ago)
11260 CRLF UTF-8 4 spaces master
```

```
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224

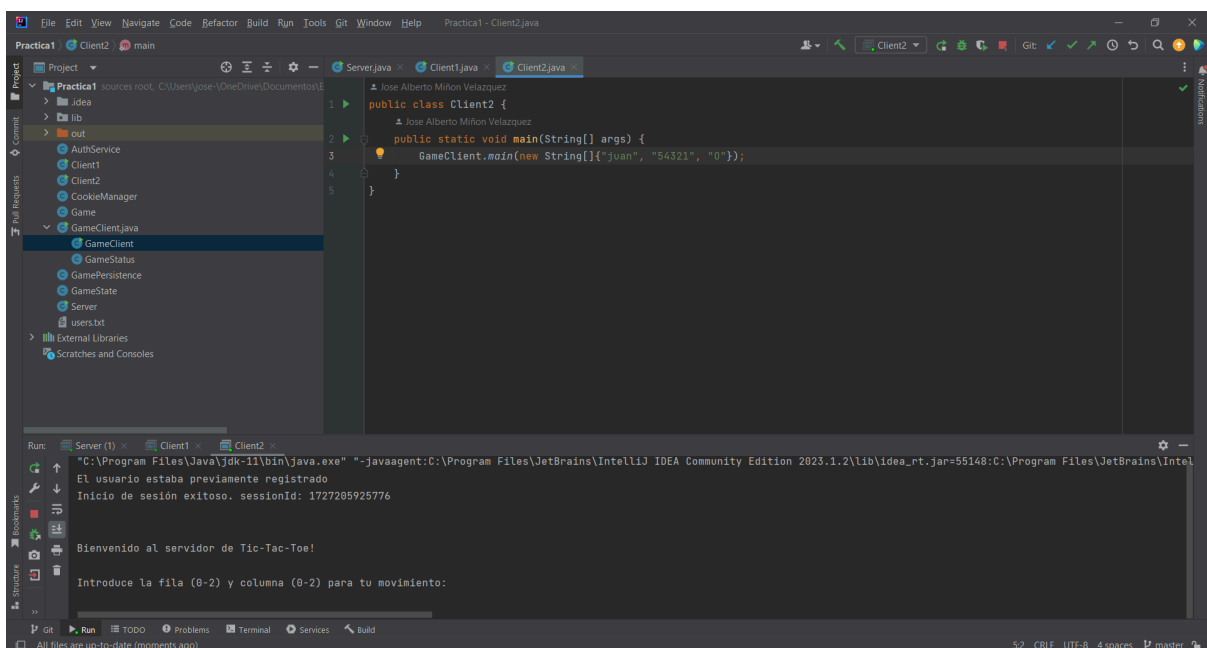
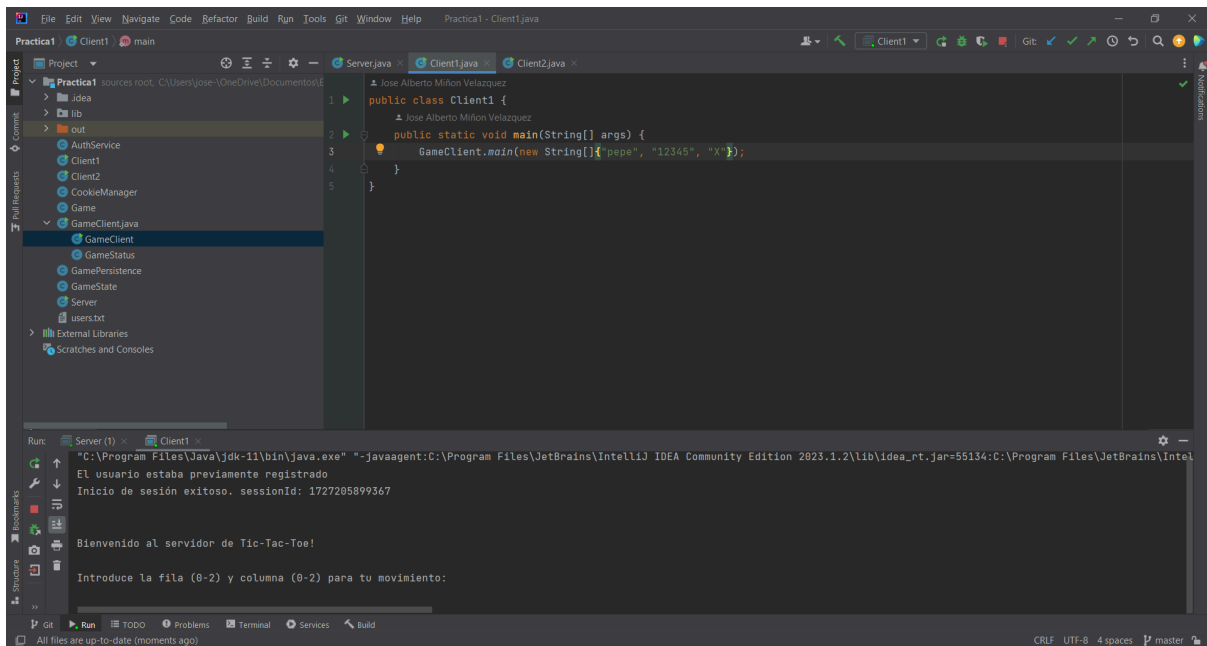
JsonObject response = new JsonObject();

if (authService.login(username, password)) {
    String sessionId = cookies.createSession(username);

    response.addProperty("status", "success");
    JsonObject dataResponse = new JsonObject();
    dataResponse.addProperty("sessionId", sessionId);
    dataResponse.addProperty("message", "Registro Exitoso");
    response.addProperty("data", dataResponse);
} else {
    response.addProperty("status", "error");
    JsonObject dataResponse = new JsonObject();
    dataResponse.addProperty("message", "Credenciales incorrectas.");
    response.addProperty("data", dataResponse);
}

out.println(gson.toJson(response));
}
```

2.- Después ejecutamos las clases Client1 y Client2, en ese orden.

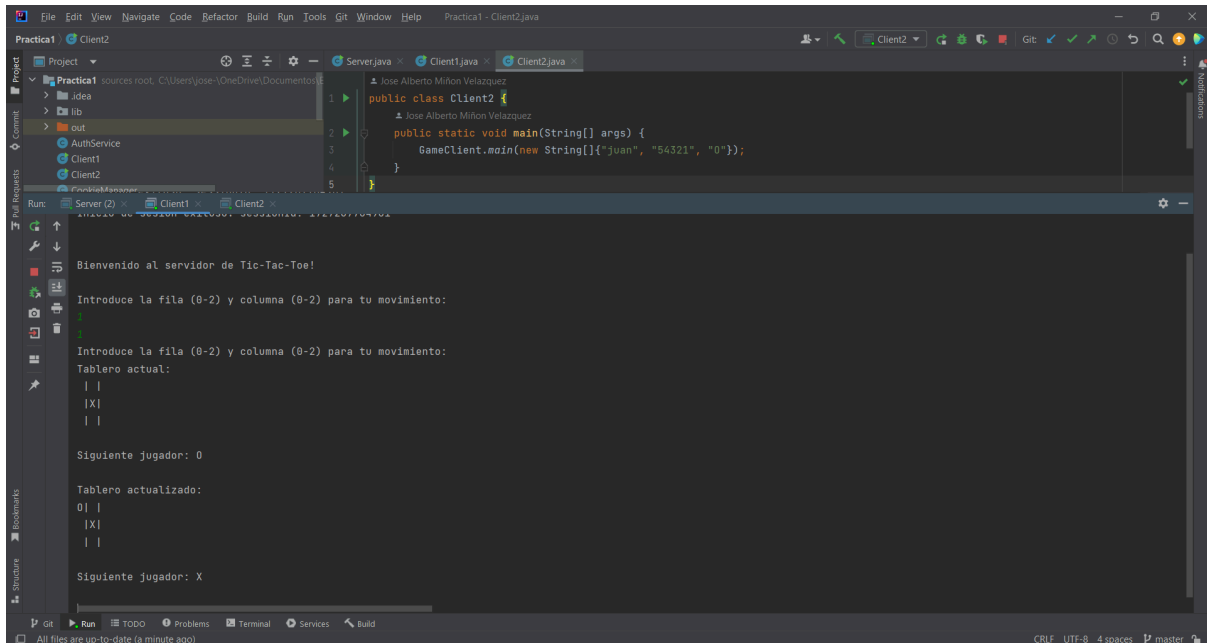


Como se ve en la imagen, en ambas terminales de IntelliJ IDEA (Client1 y Client2) donde se ejecutan los clientes, se ejecutan simultáneamente y a ambos les da la bienvenida al juego y les permite hacer un movimiento pidiendo la posición en el tablero del juego.

También podemos ver que los clientes primero intentan registrarse, pero dado que sus datos ya se encuentran almacenados de manera persistente en el archivo *users.txt*, el servidor devuelve un mensaje aclarando esto (el primer println que se muestra), después

,ambos clientes inician sesión exitosamente y el servidor proporciona un *sessionId* para la comunicación entre ambos.

3.- Ahora ingresamos la posición en el cliente 1 y en el 2.



The screenshot shows the IntelliJ IDEA interface with the `Client2.java` file open. The code defines a `Client2` class with a `main` method that calls `GameClient.moin` with arguments `new String[]{"juan", "54321", "0"};`. The Run console at the bottom shows the output of the `Server (2)` process, which is a Tic-Tac-Toe server. The output includes a welcome message, a prompt for a move, the current board state, and the next player to move.

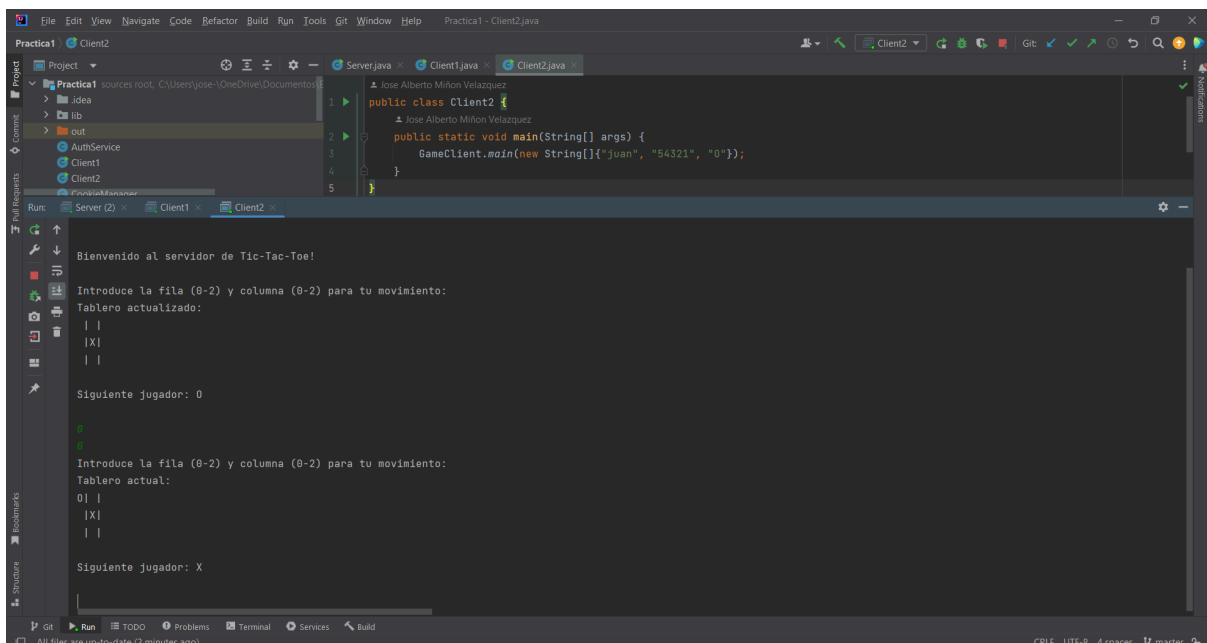
```
public class Client2 {
    public static void main(String[] args) {
        GameClient.moin(new String[]{"juan", "54321", "0"});
    }
}
```

```
Bienvenido al servidor de Tic-Tac-Toe!
Introduce la fila (0-2) y columna (0-2) para tu movimiento:
Introduce la fila (0-2) y columna (0-2) para tu movimiento:
Tablero actual:
| |
|X|
| |

Siguiente jugador: 0

Tablero actualizado:
0| |
|X|
| |

Siguiente jugador: X
```



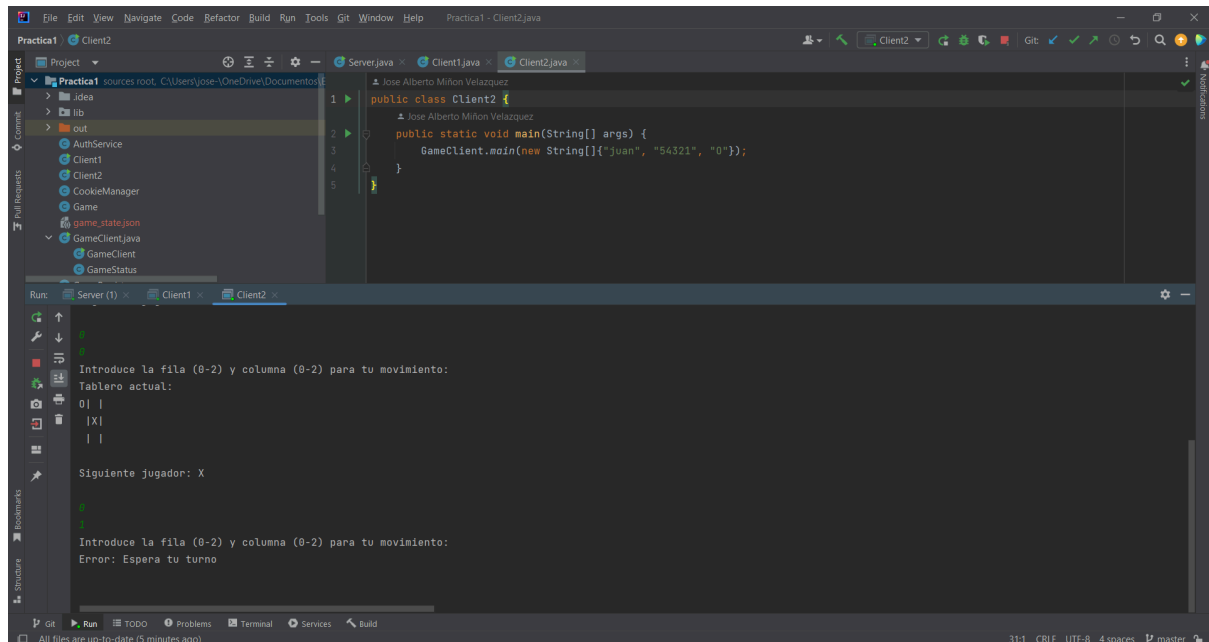
This screenshot is similar to the previous one, showing the same code and console output. The console output is identical, showing the server's response to the client's move. The status bar at the bottom indicates that all files are up-to-date (2 minutes ago).

En la terminal, se va mostrando el jugador que sigue y el tablero que resulta después de cada movimiento en tiempo real.

4.-Después de aquí hay un comportamiento medio extraño que ya no pude checar, es debido a la sincronía ya que se están usando hilos y ciclos al mismo tiempo, pero esto no afecta al juego. Simplemente hay que ingresar una nueva posición, ingresando un número a

la vez siempre, aunque el sistema no te muestre el mensaje pidiéndote los datos (se muestra después de ingresar los datos este print xd). Ya no pude arreglar esto por falta de tiempo, pero no afecta al juego.

Ahora, si algún cliente intenta ingresar una posición cuando no es su turno, el servidor no se lo permitirá, de igual forma pasará si intenta ingresar una posición inválida o ya ocupada.

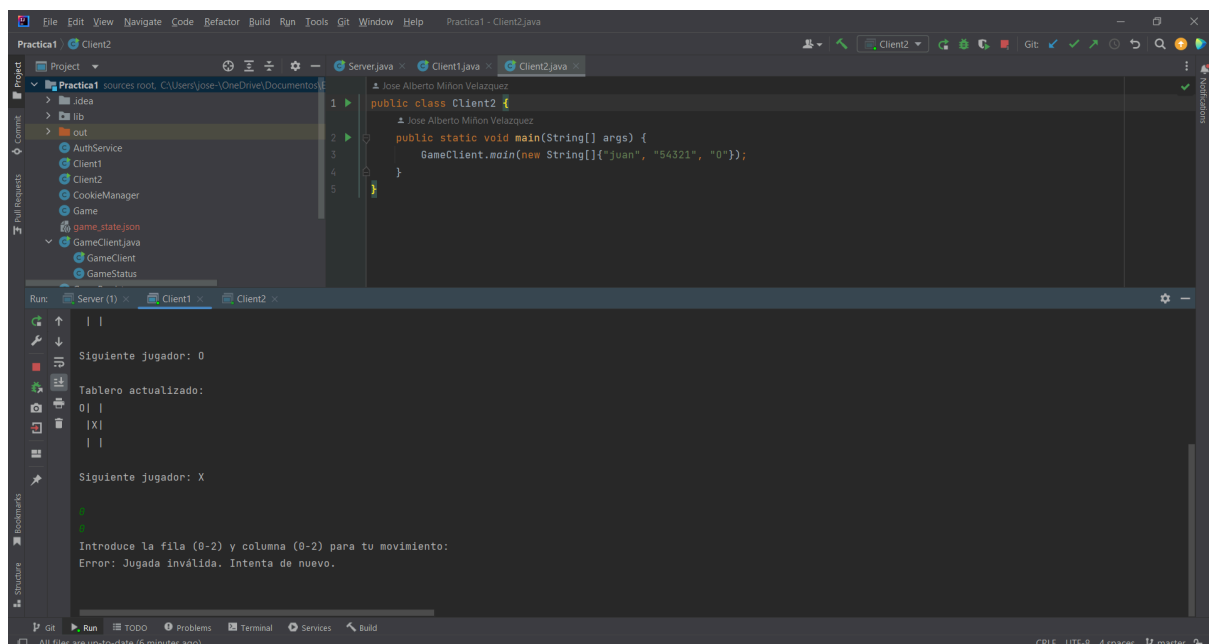


The screenshot shows the IntelliJ IDEA interface with the 'Practica1 - Client2.java' window. The code in the editor is as follows:

```
1 public class Client2 {  
2     Jose Alberto Miñon Velazquez  
3     public static void main(String[] args) {  
4         GameClient.main(new String[]{"juan", "54321", "0"});  
5     }  
6 }
```

The Run console at the bottom shows the following output:

```
Introduce la fila (0-2) y columna (0-2) para tu movimiento:  
Tablero actual:  
0 | |  
| X |  
| |  
Siguiente jugador: X  
  
Introduce la fila (0-2) y columna (0-2) para tu movimiento:  
Error: Espera tu turno
```



The screenshot shows the IntelliJ IDEA interface with the 'Practica1 - Client2.java' window. The code in the editor is the same as in the previous screenshot:

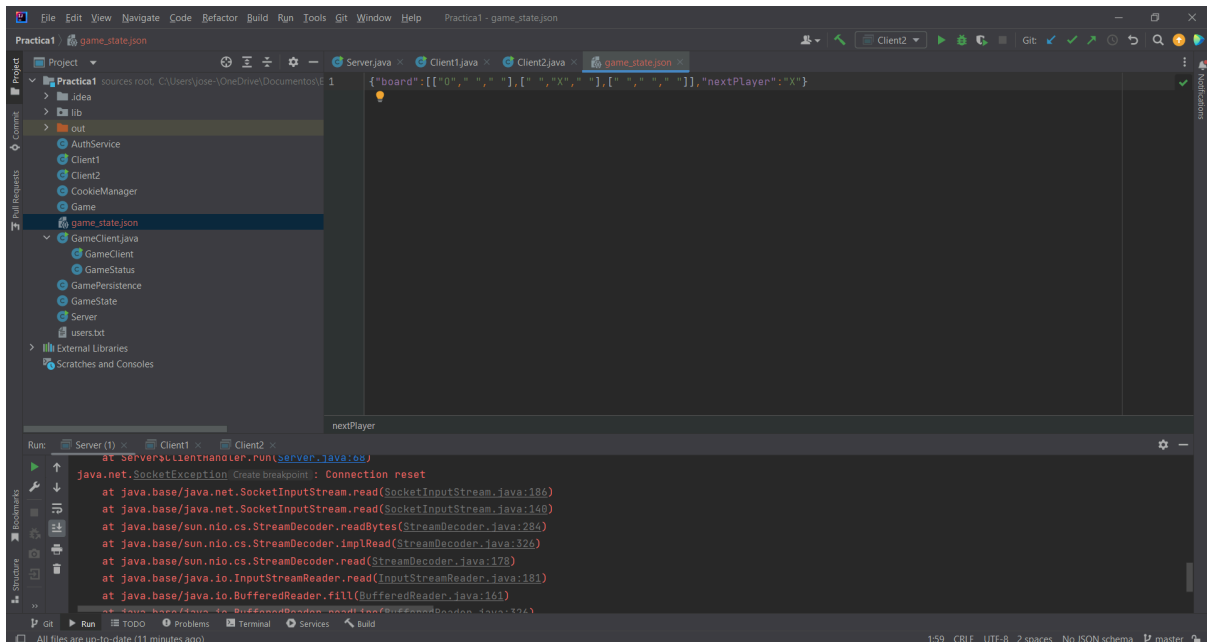
```
1 public class Client2 {  
2     Jose Alberto Miñon Velazquez  
3     public static void main(String[] args) {  
4         GameClient.main(new String[]{"juan", "54321", "0"});  
5     }  
6 }
```

The Run console at the bottom shows the following output:

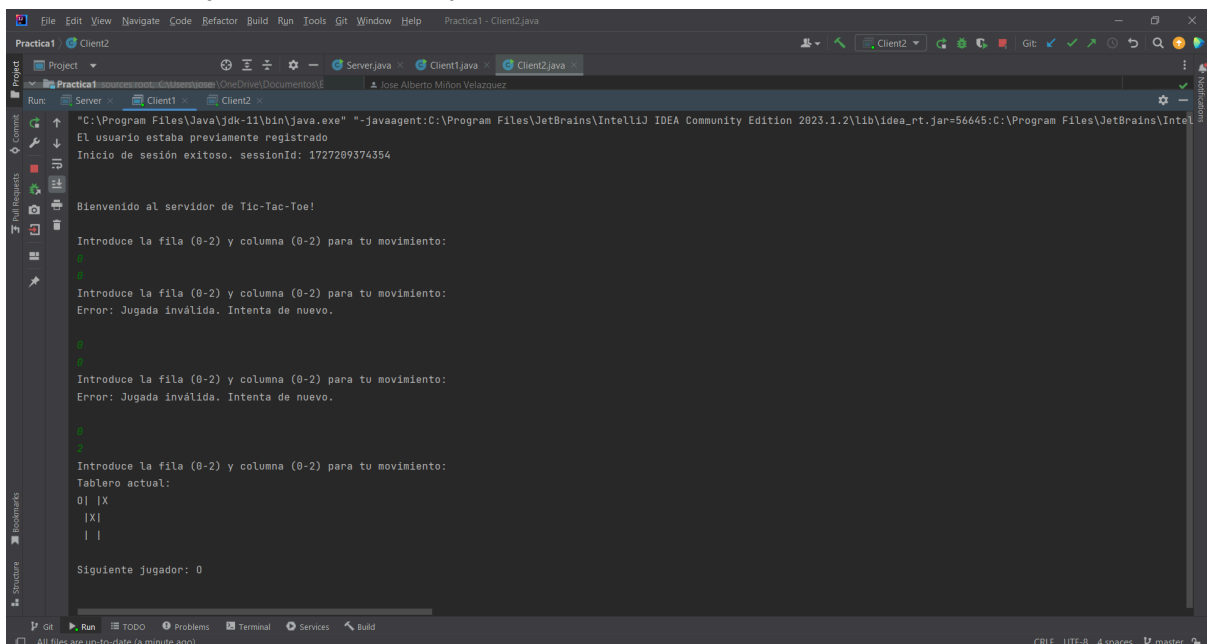
```
| |  
Siguiente jugador: 0  
Tablero actualizado:  
0 | |  
| X |  
| |  
Siguiente jugador: X  
  
Introduce la fila (0-2) y columna (0-2) para tu movimiento:  
Error: Jugada inválida. Intenta de nuevo.
```

Aquí, intentamos hacer un movimiento desde el cliente 2, pero dado que no era su turno, el sistema no se lo permitió y le mostró un mensaje, de igual forma, el cliente uno intentó ingresar una posición ya ocupada y pasó lo mismo, no lo dejo.

Si paramos el servidor en este punto, el estado del juego se registrara persistentemente en un archivo llamado *game_state.js* que se creara después de parar el servidor.



Si volvemos a ejecutar el servidor y los clientes, se empezará en el estado que se quedó



Al final, se muestra un mensaje de que hubo un ganador y quien fue, si es que hubo ganador, de lo contrario se muestra un empate.

```
Practical1 - Client2.java

Project
  Practical1
    sources
      root
        Client2.java

Run: Server Client1 Client2

"C:\Program Files\Java\jdk-11\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.1.2\lib\idea_rt.jar=56668:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.1.2\bin" -Didea.config.path=C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.1.2\conf -Didea.copyright.path=C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.1.2\copyright -Didea.home.path=C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.1.2\bin -Didea.platform.prefix=Java -Didea.vendor.id=idea -Didea.version=2023.1.2 -jar C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.1.2\bin\idea_rt.jar=56668:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.1.2\bin

El usuario estaba previamente registrado
Inicio de sesión exitoso. sessionId: 1727289383657

Bienvenido al servidor de Tic-Tac-Toe!

Introduce la fila (0-2) y columna (0-2) para tu movimiento:
Tablero actualizado:
0| I X
| X |
| I |

Siguiente jugador: 0

Introduce la fila (0-2) y columna (0-2) para tu movimiento:
Tablero actual:
0| I X
| X |
| I 0

Siguiente jugador: X
```

```
Practical1 - Client1.java

Project
  Practical1
    sources
      root
        Client1.java
        Client2.java

Run: Server Client1 Client2

Introduce la fila (0-2) y columna (0-2) para tu movimiento:
Tablero actual:
0| I X
| X |
| I |

Siguiente jugador: 0

Tablero actualizado:
0| I X
| X |
| I 0

Siguiente jugador: X

Introduce la fila (0-2) y columna (0-2) para tu movimiento:
Tablero actual:
0| I X
| X |
X| I 0

Siguiente jugador: 0

¡El ganador es: X!
```