UNIVERSIDAD DE BURGOS

DISEÑO Y MANTENIMIENTO DE SOFTWARE

DESCRIPCIÓN DETALLADA DE LA ARQUITECTURA Y DISEÑO ESCOGIDOS

Diana Bringas Ochoa
Lisa Cané Sáiz
Willow Maui García Moreno
Julen Rostan Saez
Rute Catarina Teixeira Dos Santos

Contenido

Introducción

Partiendo de la anterior práctica replanteamos el diseño para que cumpliese el principio SOLID de **responsabilidad única**, ya que en el diseño anterior diseño se mezclaban las responsabilidades de gestión de los mensajes (comunicación) con la lógica de juego o presentación. Era el caso de las clases *InterfazJugador* y *Arbitro*, que en sus métodos *jugar* y *arbitrar* respectivamente en función de los códigos de mensaje recibidos llamaban a las demás funciones de la clase.

Para que cada módulo tuviese una única responsabilidad añadimos dos clases más al diseño, una al servidor y otra al cliente, que funcionan como intermediarios que hacen la gestión de los mensajes llamando a las funciones respectivas del árbitro (en caso del servidor) o de la interfaz del jugador (en caso del cliente), para que esta responsabilidad no recaiga en dichas clases.

Al cumplir este principio SOLID, cuando sea necesario hacer un cambio en la interfaz gráfica o en las comunicaciones, ya no habrá que modificar todas las clases, sino únicamente la correspondiente a la capa de presentación o a la capa de comunicaciones.

Además, para no tener código repetido, creamos una carpeta ***Biblioteca*** que contiene a las clases que utilizan el cliente y el servidor.

Hemos añadido también una clase ***Pieza*** para que el juego tenga mejor escalabilidad a la hora de poder introducir en un futuro otros juegos que no necesariamente empleen las mismas piezas que el Tres en Raya. Y de esta forma evitamos el uso de tipos primitivos, empleando tipos propios de nuestro modelo de datos.

Como ahora la aplicación contiene dos juegos distintos, hemos aplicado el **Patrón de diseño Estrategia** al árbitro ya que deberá comportarse de forma distinta en ciertos métodos en función de si se juega al Tres en Raya o al Conecta 4. Por ejemplo, en el Tres en Raya las piezas se pueden colocar en la posición que se quiera (si no están ocupadas y están dentro del tablero) ya que se trata de un tablero plano, en cambio en el Conecta 4 las piezas se dejan “caer” en la columna elegida (si no está completa y está dentro del tablero) ya que es un tablero vertical, por lo que las comprobaciones de los movimientos serán distintas; o en el método de comprobación de fin de juego (*esFin*) al tratarse de tableros de distinto tamaño habrá que comprobar distinto número de filas y columnas.

Comunicación entre Cliente y Servidor

Para la comunicación entre Cliente y Servidor hemos escogido utilizar **sockets**.

Los socket están definidos por la IP de la máquina, el puerto que escucha y el protocolo utilizado. Es por eso que en el método de inicialización de Cliente y Servidor lo primero que hacemos es instanciar el socket, establecer el host (IP de la máquina) y el puerto (el que escucha). En nuestro caso empleamos sockets de flujo (SOCK_STREAM) con protocolo INET (AF_INET).

Los códigos de mensaje que emplea el Cliente para enviar al Servidor son:

- **100**: indica el fin de la partida, para cerrar de esta forma la conexión del cliente.
- **101**: indica que la clase *InterfazJugador* ya le ha mostrado el tablero al jugador del turno actual, por lo que el Servidor le indica al Cliente que deberá solicitarle el movimiento al jugador del turno actual.

- **102**: indica que un cliente quiere jugar una partida, enviándose además como objeto el número de jugador establecido (1 o 2) en función de si ha sido el primero en conectarse o el segundo.
- **103**: indica que se quiere dibujar el tablero actualizado.
- **104**: indica que se ha enviado el movimiento que el jugador quiere realizar, el movimiento está contenido como objeto.
- **105**: indica que el tablero se ha actualizado, es decir, se ha colocado la pieza en el tablero ya que el movimiento introducido por el jugador es correcto.
- **1**: indica que el juego seleccionado es el Tres en Raya.
- **2**: indica que el juego seleccionado es el Conecta 4.

Los códigos de mensaje que emplea el Servidor para enviar al Cliente son:

- **200**: indica el fin de la partida, para mostrarle al jugador el resultado de la partida.
- **201**: indica que se inicializa el jugador con el número que se le ha asignado (1 o 2).
- **202**: indica que se ha devuelto el tablero actualizado, para que la interfaz del jugador imprima el tablero.
- **203**: indica que se tiene que volver a solicitar el movimiento ya que el que se ha introducido anteriormente era incorrecto.
- **204**: indica que el movimiento que se ha introducido es correcto.

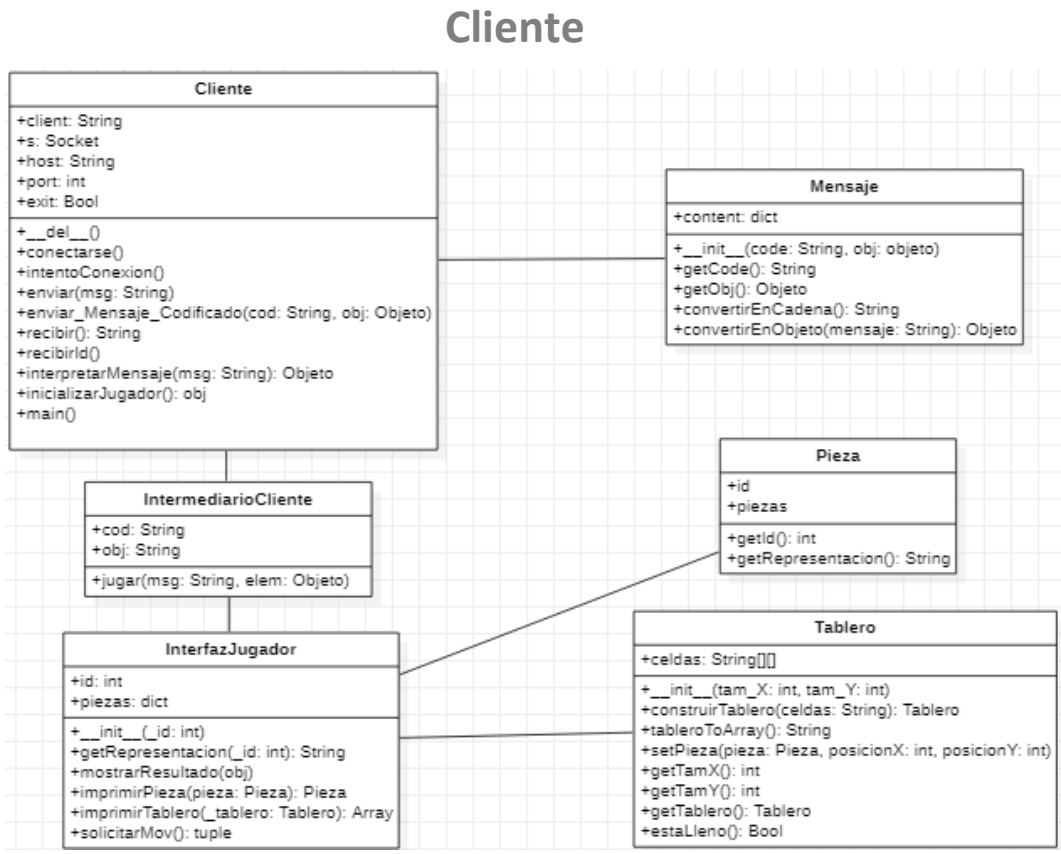

Ilustración 1 Diagrama de clases del Cliente

Clase Cliente

La clase *Cliente* pertenece a la **Capa de Comunicaciones**, es la encargada de comunicarse con el Servidor a través de mensajes codificados, que es quién contiene los datos necesarios para jugar (tablero, piezas, etc).

Establece la conexión e interacciona con los jugadores (para jugar) a través de la clase ***IntermediarioCliente***.

Se encarga de la conversión a objetos de los mensajes recibidos por parte del Servidor, y de la conversión a cadena de los mensajes que se quieren enviar al Servidor.

Clase IntermediarioCliente

La clase *IntermediarioCliente* pertenece a la **Capa de Comunicaciones**, se encarga de interpretar el código de los mensajes recibidos para llamar a la función que corresponda de la clase ***InterfazJugador***, devolviendo el código y objeto de mensaje que se tienen que enviar al Servidor.

Clase InterfazJugador

La clase *InterfazJugador* pertenece a la **Capa de Presentación**, se encarga de mostrar (presentar) la representación del tablero y las piezas, y de mostrar los mensajes correspondientes al jugador.

Cuando sea el turno del jugador, le muestra el tablero actualizado (que ha obtenido del Árbitro a través del Servidor) le solicita el movimiento que desea realizar.

Servidor

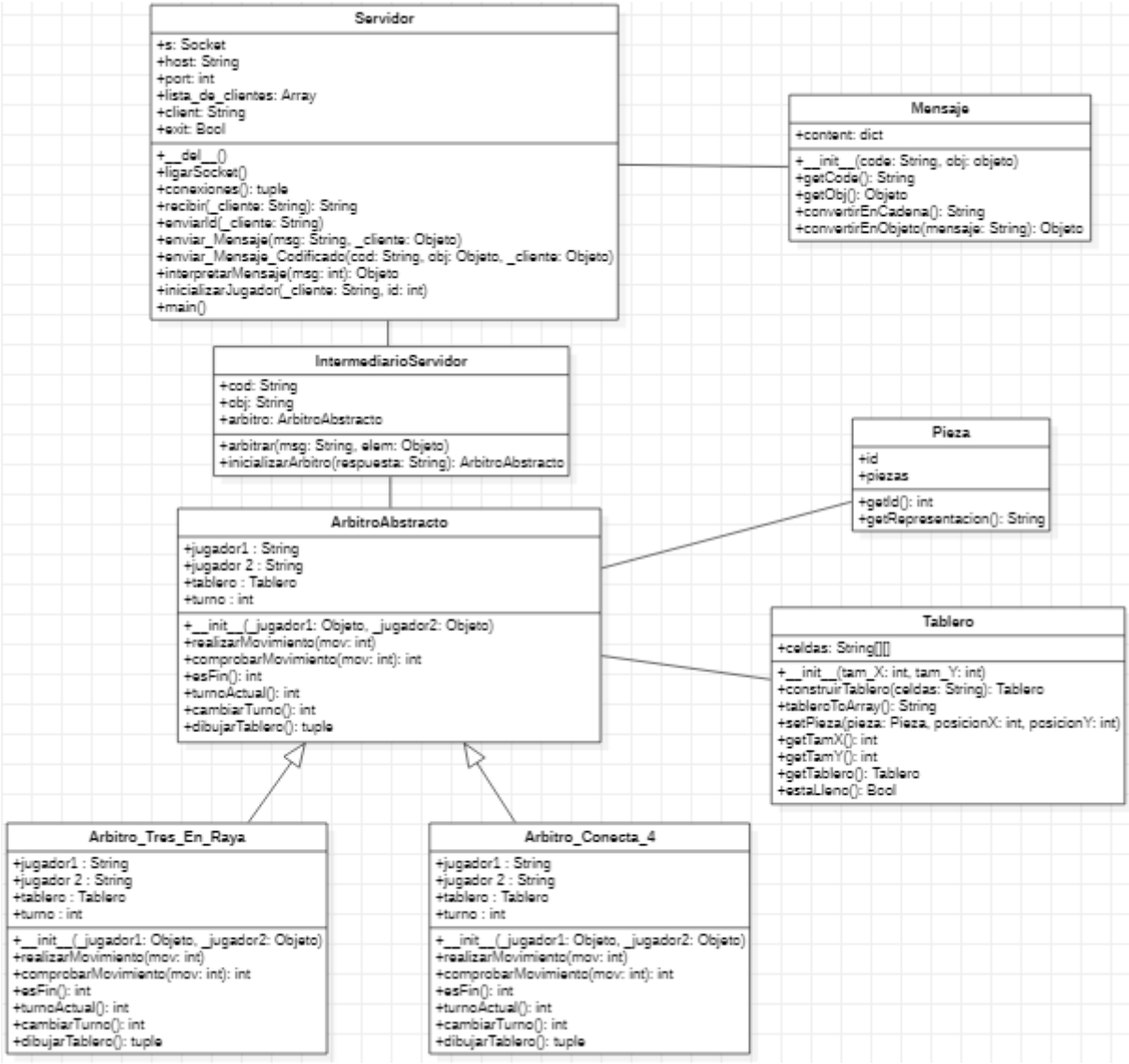

Ilustración 2 Diagrama de Clases del Servidor

Clase Servidor

La clase *Servidor* pertenece a la **Capa de Comunicaciones**, es la encargada de comunicarse con el Cliente a través de mensajes codificados, para proporcionarle los datos necesarios para jugar (tablero, piezas, etc).

Establece la conexión e interacciona con el árbitro en concreto (en función del juego seleccionado) a través de la clase ***IntermediarioServidor***.

Se encarga de la conversión a objetos de los mensajes recibidos por parte del Cliente, y de la conversión a cadena de los mensajes que se quieren enviar al Cliente.

Arbitro

El árbitro implementa el patrón de diseño Estrategia, de esta forma se tienen tres clases, el abstracto que define las funciones de los otros dos que tienen el comportamiento concreto de cada juego (Tres en Raya y Conecta 4).

El árbitro pertenece a la **Capa de la Lógica del Juego**, ya que es quién contiene las reglas que se deben tener en cuenta a la hora de jugar.

Una clase ***ArbitroAbstracto*** que tiene implementadas las funciones que sirven tanto para el ***Arbitro_Tres_En_Raya*** y para el ***Arbitro_Conecta_4*** y tiene declaradas las funciones que tienen que implementar de forma concreta los árbitros en función del juego seleccionado.

Una clase ***Arbitro_Tres_En_Raya*** para cuando se selecciona el juego del Tres en Raya y una clase ***Arbitro_Conecta_4*** para cuando se selecciona el juego del Conecta 4, en las que se redefine la inicialización, la comprobación y realización de los movimientos y la comprobación del fin de juego.

Clase IntermediarioServidor

La clase *IntermediarioServidor* pertenece a la **Capa de Comunicaciones**, se encarga de interpretar el código de los mensajes recibidos para llamar a la función que corresponda de la clase ***Arbitro*** (en función del juego al que se esté jugando se llama la del tres en raya o al del conecta 4), devolviendo el código y objeto de mensaje que se tienen que enviar al Cliente.

Biblioteca

Esta carpeta contiene las clases usadas tanto por el Servidor como por el Cliente.

Clase Mensaje

La clase *Mensaje* pertenece a la **Capa de Comunicaciones**, se encarga de encapsular y desencapsular los mensajes y objetos de la comunicación.

Clase Pieza

La clase *Pieza* pertenece a la **Capa de Datos**, contiene el id y la representación de las piezas.

Clase Tablero

La clase *Tablero* pertenece a la **Capa de Datos**, contiene el tablero de la partida, el cual se irá actualizando, se podrá obtener, imprimir por pantalla y comprobar si está lleno.

En función del juego seleccionado tendrá un tamaño u otro, el cual se debe especificar en el método de inicialización.