Consideraciones Generala

Inicio de Conexión: Utilizaremos RPC para la creación de una partida entre los nodos. Como queremos hacer transparente la conexión entre ellos (que no se deban intercambiar las IPs manualmente) decidimos utilizar un nodo específico que hará de intermediario en la gestión de la dinámica del juego. El mismo será llamado **nodo Mesas.**

Nodo Mesas:

- **Inicio Partida (RPC):**
  - la IP del mismo es conocidad por todos los nodos jugadadores, pero está embebida en el código, por lo que será transparente para el usuario (transparencia de conexión).
  - Cuando un jugador quiere crear una mesa, deberá solicitarlo al nodo Mesas, enviándole su IP junto con la cantidad de jugadres que formarán dicha mesa. Esto se realizará a partir de RPC con el mensaje **crearMesa(IP, cantJugadores)**.
  - Cuando un jugador quiere unirse a una mesa deberá solicitarlo al nodo Mesas, enviándole su IP. Esto se realizará a partir de RPC con el mensaje **unirseMesa(IP)**.
  - Al haber creado satisfactoriamente una mesa bajo pedido se quedará esperando a llenarla con la cantidad especificada. Una vez que este llena le responderá a todos los jugadores que se conectaron con un **empezarPartida(cantJugadores)**. Hasta que esto no ocurrá, tanto el nodo que solicita crear la mesa como el que quiere unirse no recibirá respuesta del nodo Mesa, permaneciendo bloqueado.
- **Arranca el Juego (Sockets):**
  - para el desarrollo del juego, utilizaremos Sockets debido a que utilizaremos un sistema por turnos manejado por el nodo Mesas.
  - Según el orden en que se hayan conectado los jugadores a la mesa, será el orden que se manejarán los turnos.
  - Cuando es el turno de un jugador, se le enviará un mensaje **jugarTurno().** Una vez que el turno acabe el nodo jugador responderá con un **termineTurno(puntajeActualJugador)** al nodo Mesas (bloqueandose al finalizar su turno). Luego el nodo Mesas otorgará el turno al siguiente jugador.
- **Enviar Dado (Sockets):**
  - El servidor mesa será el encargado de generar la secuencia aleatoria de dados y deberá enviarle a todos los nodos el valor de los 5 dados.
    - Los que no estén jugando sólo mostrarán en pantalla los dados recibidos.
    - El jugador que esté jugando deberá determinar que dados se queda y solicitar al nodo Mesa que dados volver a tirar. El nodo Mesa envía a todos los nodos **valoresDados(d1,d2,d3,d4,d5)** y el jugador actual contesta con **pedirDados(p1,p2,p3,p4,p5),** donde di son los valores de los dados y pi es un flag [1,0] indicando si se vuelve a tirar o no el dado. Finalmente el nodo Mesa vuelve a mandar a todos los nodos los nuevos valores de los dados (y cada nodo compara para saber que dados cambiaron).
- **Fin del juego (Sockets):**
  - como el nodo Mesas conoce la cantidad de turnos que se jugaron, al terminar el último turno del último jugador tendrá los puntajes finales de todos los jugadores. Por lo tanto, enviará a cada uno de los jugadores el mensaje **ganador()** o **perdedor(puntajeGanador)** según haya ganado o perdido dicho jugador, siendo puntajeGanador el puntaje del jugador que gano la partida.