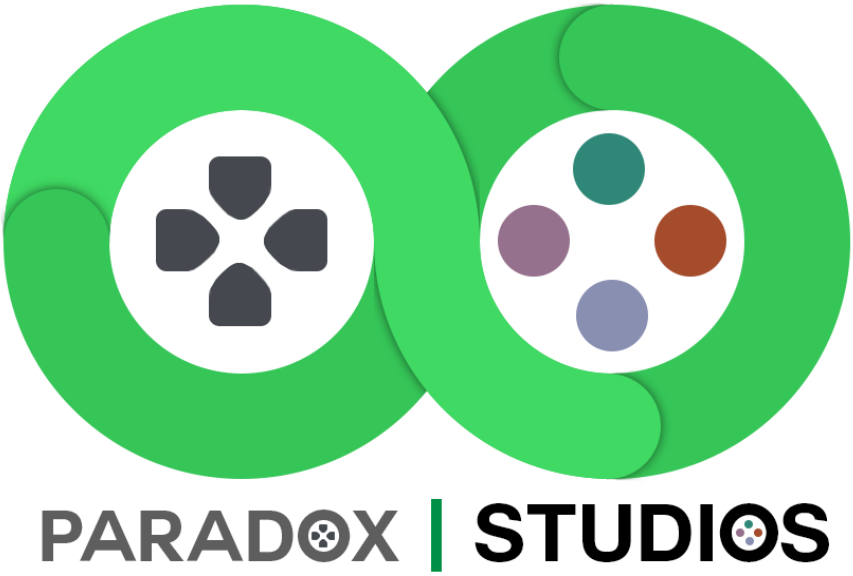
Diseño de requerimientos y funciones de red



Paradox Studios:
Moltó Ferré, Enrique
Muñoz Periñán, José Luis
Pérez Cristo, Rubén
Rebollo Berná, Antonio
Zamora Pastor, Julio

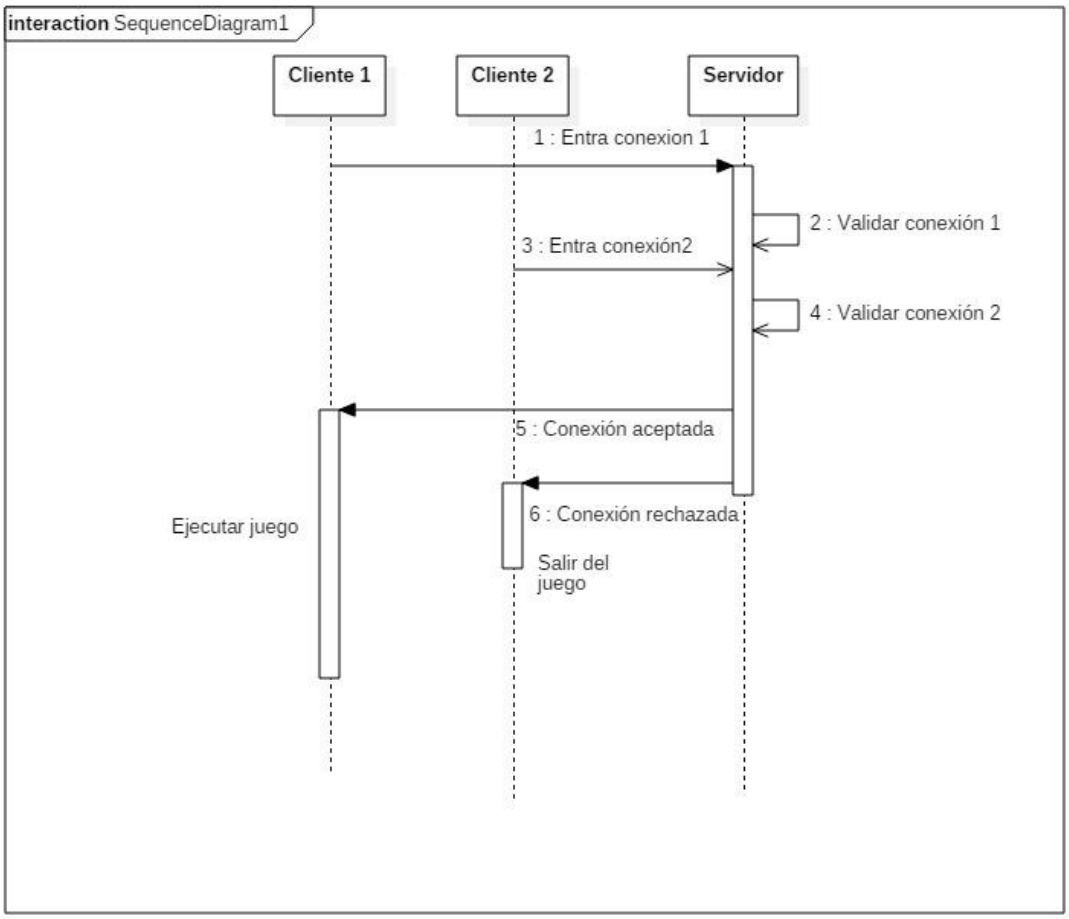
Diseño de requerimientos y funciones de red

Nuestro motor de red tendrá los siguientes requerimientos:

- Ejecutar el servidor correctamente
- Permitir que varios clientes se conecten a un servidor.
- Permitir paso de mensajes entre el cliente y servidor y viceversa.
- Permitir una conexión estable entre clientes y un tiempo de respuesta bastante bueno.
- El cliente será capaz de buscar entre los servidores abiertos disponibles y conectarse al que el desee.

Nuestro motor de red tendrá las siguientes funciones:

- Permitir que varios clientes se conecten a un servidor.
    - El cliente manda un mensaje de conexión al servidor y este le responde si su conexión ha sido aceptada o no. Si la conexión es aceptada entonces se crearía el player en el cliente y en el servidor.

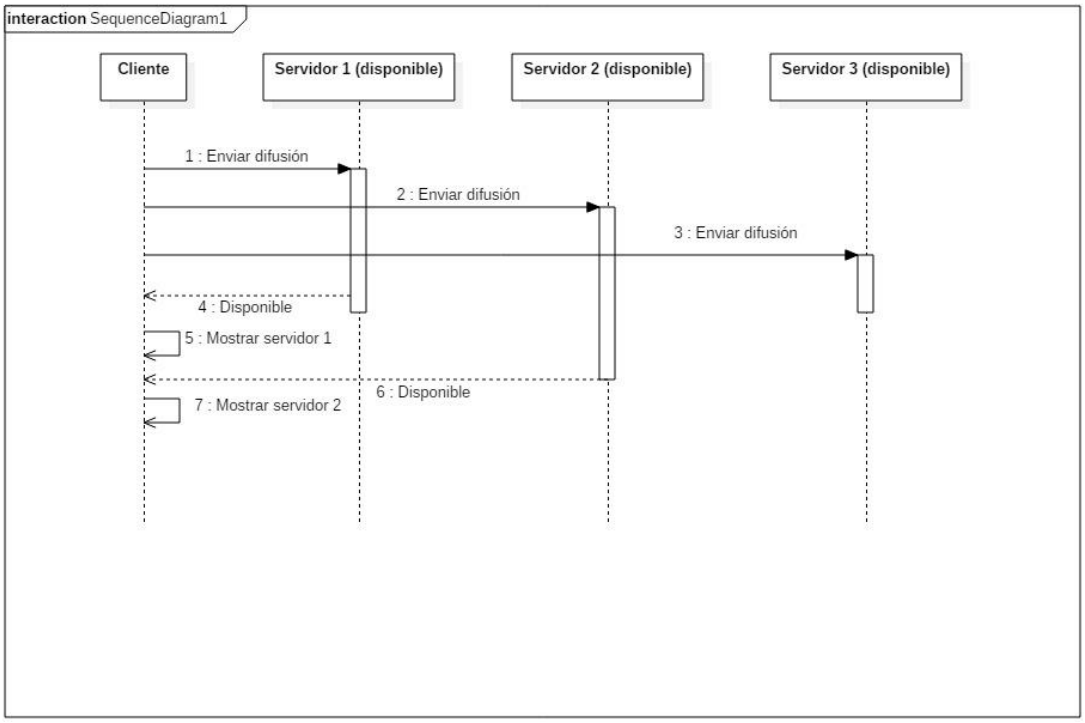


- Permitir paso de mensajes entre el cliente y servidor y viceversa.

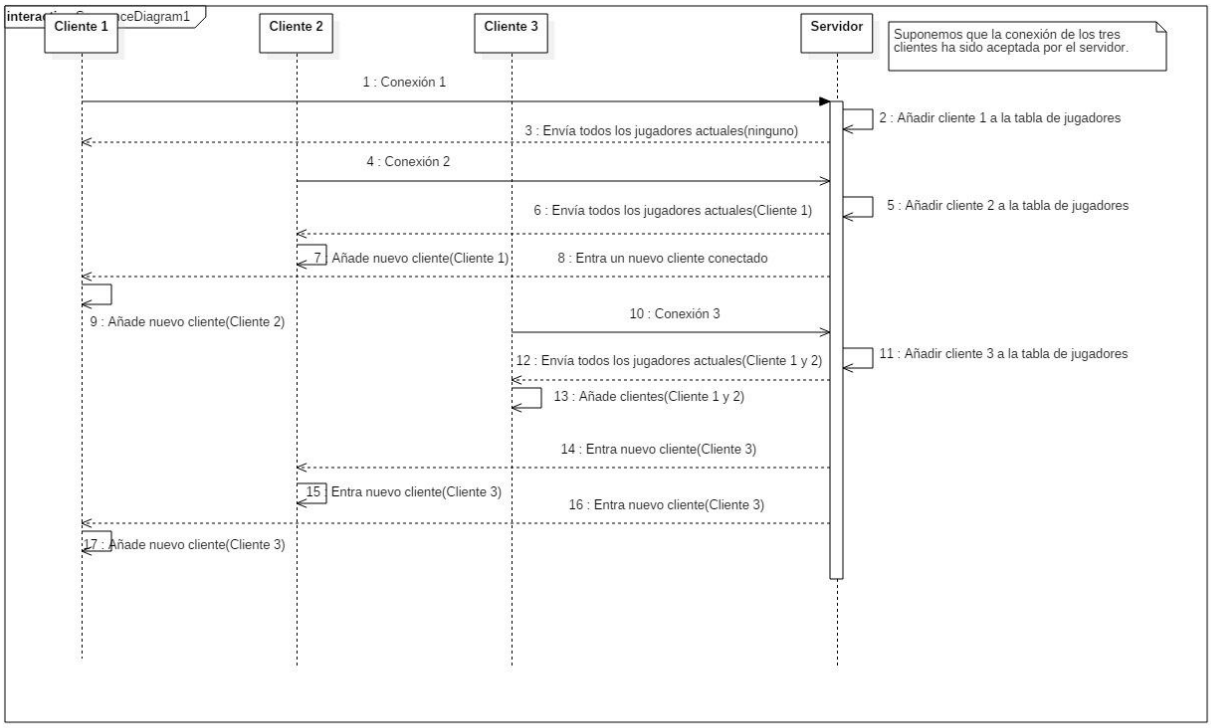
- El servidor y el cliente se enviaran paquetes entre ellos usando el nodo de conexión y sabiendo quien es cada uno gracias a su GUID que los identifica.

- El cliente será capaz de buscar entre los servidores abiertos disponibles y conectarse al que el desee
    - Esto se realiza mediante un ping a broadcast (255.255.255.255) y los servidores abiertos responden y el cliente que lanzo el ping se guarda todas las ip de los servidores que respondieron y los muestra en una lista. Una vez hecho esto podrás elegir entre los servidores abiertos para poder conectarse a uno de ello.

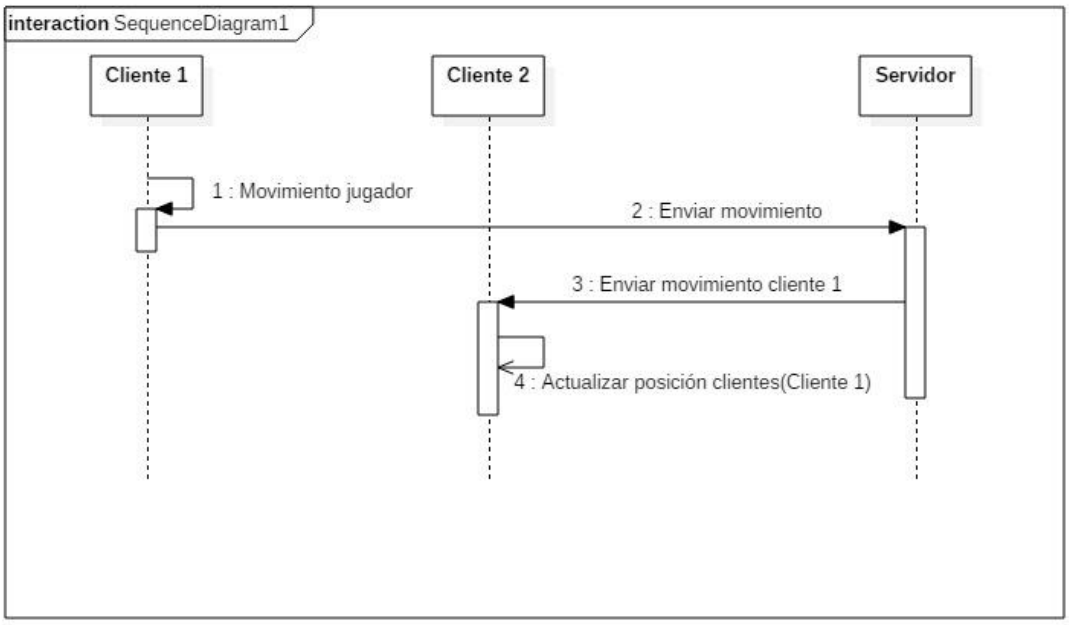
- Visualizar todos los clientes que hayan conectados en ese momento



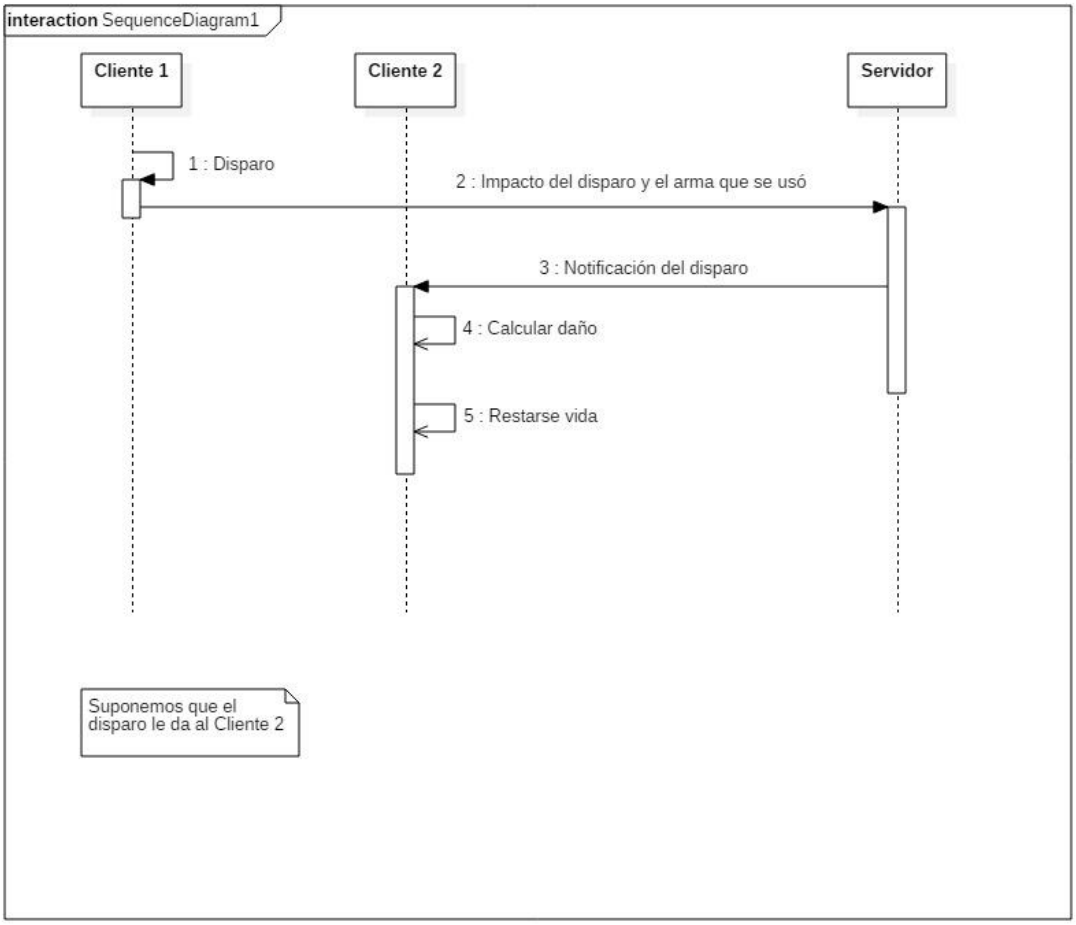
    - Esto se hará gracias al mapa de enemigos que tiene cada cliente ya que cada enemigo es un entity y por lo tanto tiene un nodo grafico que se dibujara por pantalla, cada vez que se conecte un nuevo jugador se añadirá este al mapa de jugadores y por tanto empezara a visualizarse por pantalla.



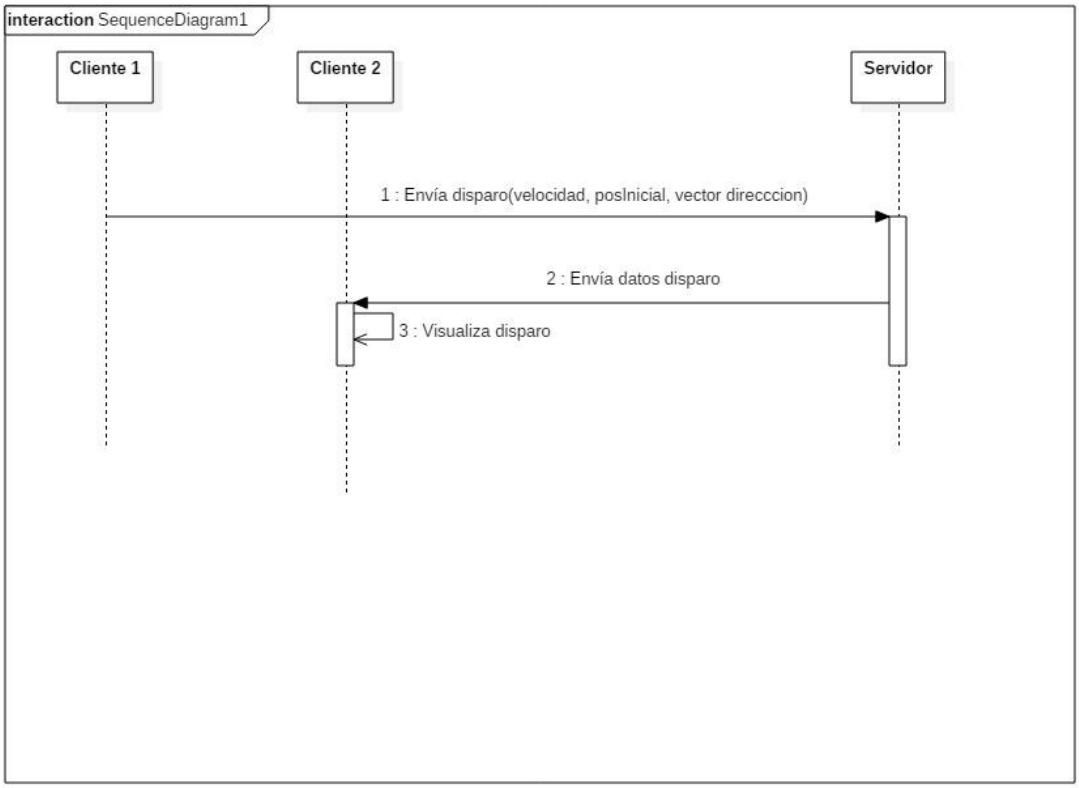
- Ver el movimiento y la rotación de estos en cada momento.
    - Como tenemos una lista de enemigos en cada clientes con todos sus componentes también tenemos también su posición que es donde se visualizan, entonces en cada update del jugador este envía su nueva posición al servidor y este se encarga de enviar la nueva posición de ese player a todos los clientes conectados (menos al propio player que se movio), así en cada cliente recibiríamos un paquete de movimiento con la nueva posición de cada uno de los enemigos en cada update y hacemos la interpolación de estos. Para saber de quién es cada posición usamos los GUID, ya que la posición se envía en una estructura con el GUID del player que tiene esa posición entonces cuando recibes las nuevas posiciones en un cliente obtienes mediante ese guid que te llega en el paquete el enemigo al que vas a cambiarle la posición.



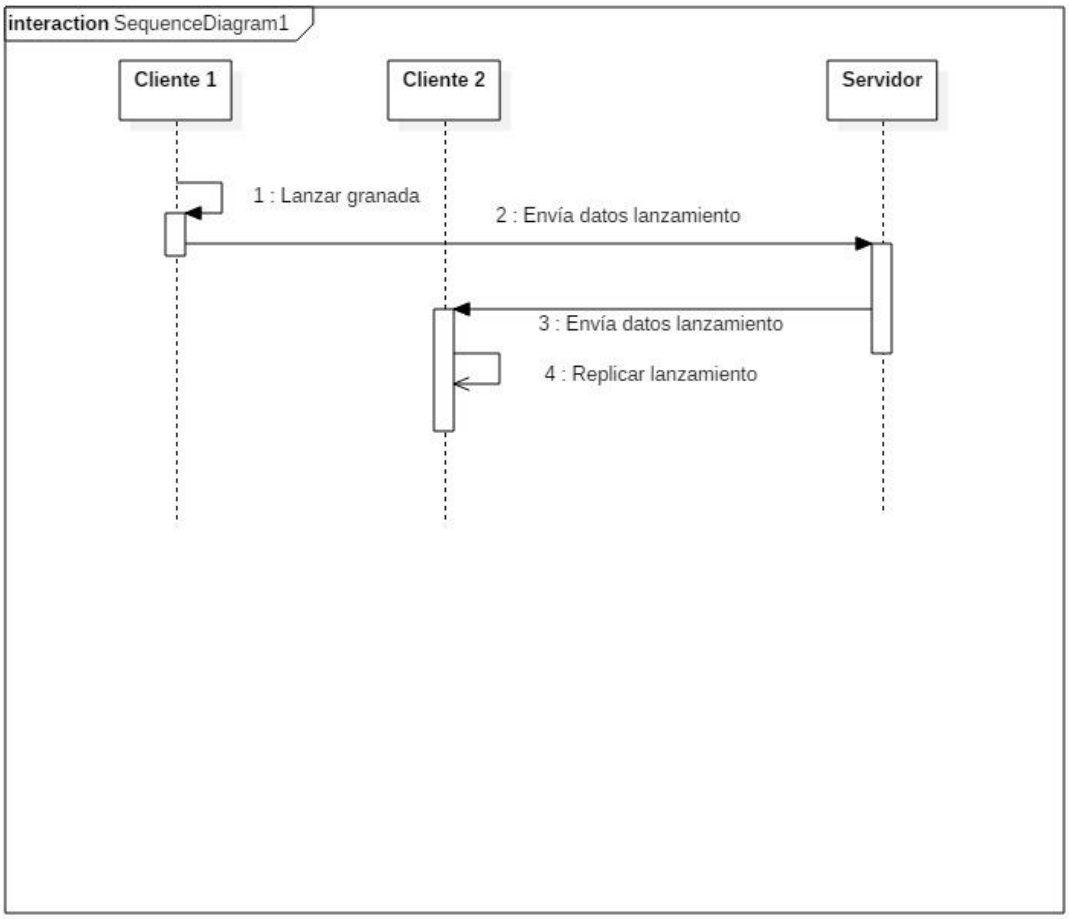
- Disparar a diferentes enemigos (clientes) y poder quitarle vida en caso de impacto.
    - Cuando un jugador dispara se lanza un raycast si este colisiona con algún enemigo entonces este notificara al servidor y el servidor notifiacara al cliente que se le ha dado para que este se reste vida. Si este se resta vida y muere se notificara al servidor para que este avise a todos los clientes que están conectados para que sepan que ese jugador ha muerto.



- Visualizar de balas en el cliente que se está ejecutando tanto de las que dispare el propio player como de las que disparen los diferentes enemigos.
    - Las balas son un típo de entities un tanto especial ya que no tienen cuerpo físico, simplemente en su constructor se le pasa un vector dirección, una velocidad y un tiempo de vida y la bala se ira updateando con estos parámetros. Así cuando un cliente dispara una bala le envía un paquete al servidor con el vector de dirección (vector cámara de ese jugador),el punto desde donde se dispara, y el tiempo de vida (se calcula mediante la velocidad de la bala y el punto de impacto que se saca mediante el raycast de la bala), este paquete se enviaría a todos los clientes conectados menos al que disparo y así estos recibirían esta información y crearían una nueva bala con esa dirección y velocidad, que aria visualizándose hasta que se pase al tiempo de vida.

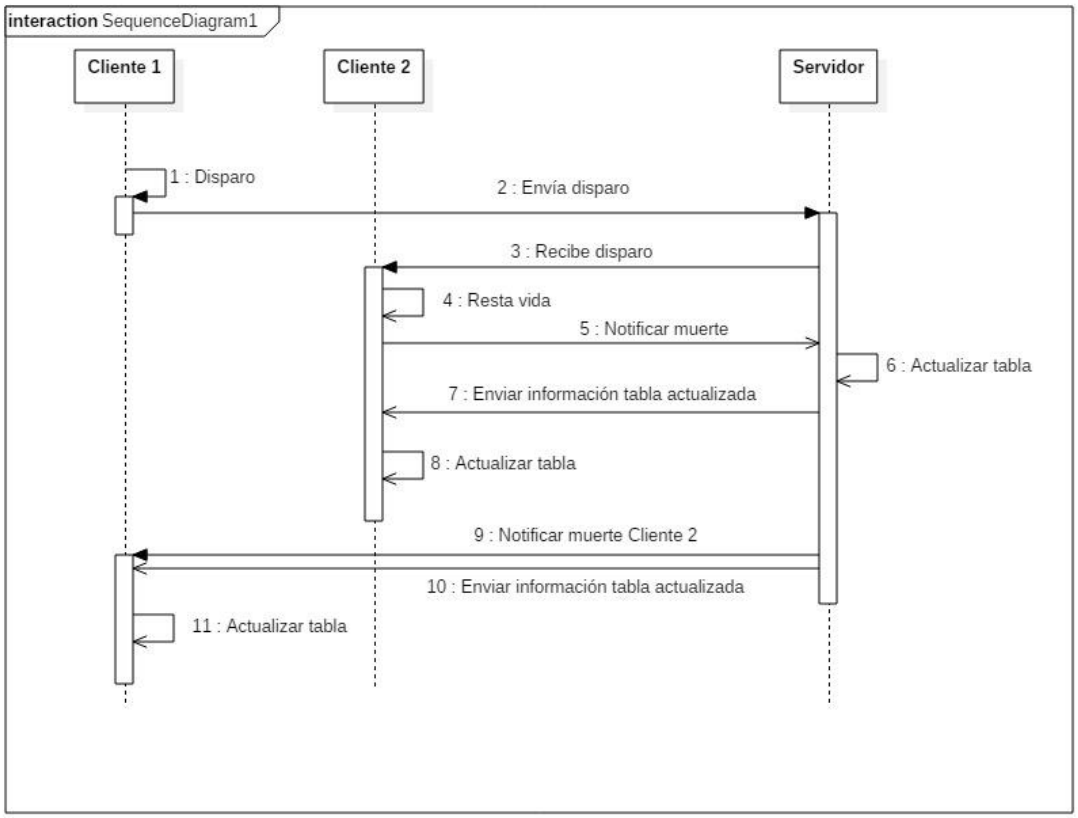


- Visualizar del rocket tanto los propios como de los enemigos.
    - En el caso del rocket es un poco distinto ya que en este caso este objeto sí que necesitaría tener colisión para realizar el radio de explosión y saber a qué enemigos daña, entonces cuando un jugador dispara un rocket este envía un mensaje al servidor con la dirección del disparo, su posición de disparo y su GUID, entonces el servidor enviaría este mensaje a todos los clientes menos al que disparo, y estos cuando lo reciben crearían un nuevo rocket desde el enemigo que disparo (lo saben gracias al GUID que viene en el paquete), la dirección que lleva y desde el punto de salida indicado, y este objeto sería una entity el cual se actualiza su posición en cada update y cuando colisione haría el radio de explosión, con el cual quitaría vida a los enemigos.
- Lanzar granadas y poder quitar vida según la cercanía de un enemigo a la granada en el momento de la explosión.
    - Cada enemigo del mapa de enemigos en un cliente tiene una granada. Cuando un cliente dispara su granada envía un paquete al servidor que es una estructura (con vector dirección y el guid del jugador que la lanzo), el servidor envía esta estructura a todos los clientes menos al que la lanzo. Cuando los clientes reciben la estructura hacen un lanzamiento de granada pero desde el enemigo que tiene el mismo guid de la estructura, así la granada se lanzaría desde el enemigo que la lanzo y el vector de dirección seria el que se le pasa por la estructura.



- Visualizar las granadas lanzadas por los enemigos.
    - Con el paquete recibido en el punto anterior se crea la granada y esta empieza a visualizarse, como sería una entity se iría actualizando en cada update.
- Detectar la muerte de un enemigo.
    - Cuando un jugador se resta vida y esta baja a 0 se envía un mensaje al servidor de que ese jugador ha muerto, para ello se le envía una estructura con 2 GUID, uno es su GUID y otro es el GUID del jugador que le mato (bien con granada, rocket, o disparo de algún arma) para que luego se pueda actualizar la tabla con esta información. El servidor enviara al resto de clientes únicamente el guid del jugador que murió para que todos sepan que ese jugador ha muerto.
- Visualizar una tabla de puntuación con todos los clientes que hayan jugando en ese momento (incluido tu) para poder ver en todo momento las bajas, muertes y puntuación de cada jugador.
    - La tabla es un unordered map cuya clave es un GUID y su valor es una estructura (TFila) que tiene el guid, nombre, bajas, muertes y puntuación de cada jugador que esté conectado al servidor.
      Cada vez que un jugador muere o mata se buscara su fila en la tabla gracias al GUID y se actualizarán los datos bien aumentando las bajas o aumentando las muertes.

- Esta tabla se actualizara tanto en el servidor como en todos los clientes cada vez que haya una baja o una muerte.
    - Cada vez que alguien muera o mate se actualizara la tabla en el servidor y además se notificara a todos los clientes de que la tabla ha cambiado y se pasaran los cambios de esta para que todos los clientes la actualicen y todos tengan la misma tabla.



- Aplicar un impulso a los enemigos que reciben el impacto de un rocket.
    - Cuando un rocket es disparado este se replica en los clientes conectados, de manera que cuando el rocket colisiona aparece un radio de explosión, si algún jugador se encuentra dentro de este radio le restara una cantidad de vida proporcional a la distancia entre el jugador y el centro del impacto.
      Además se le aplicara un impulso que será en la dirección del vector que sale del centro de la colisión y va hacia el player.
- Poder coger los paquetes de vida y weapon drops y que estos no estén disponibles en todos los clientes ya que cuando uno lo coge no están disponible. El servidor se encarga de sincronizar el tiempo de respawn de los paquetes de vida y weapon drops para que todos estén en las mismas condiciones.
    - Cuando un jugador se conecta a una partida por primera vez el servidor se encarga de decirle si los paquetes de vida y armas están disponible o si por el contrario ya han sido cogidos (porque te has unido a una partida ya empezada donde un player ya ha cogido un arma por ejemplo) en este caso el servidor envía un mensaje al nuevo cliente con el tiempo que falta hasta que reaparezca el paquete para que todos los clientes tengan así sincronizados el spawn de los pick up.

- Cuando un cliente coge algún arma y cambia entre las armas que tenga el servidor se encargara de notificar al resto de clientes para que todos vean a cada jugador con el arma que tiene actualmente.
    - Si un jugador cambia de arma por ejemplo de la pistola al fusil de asalto este envía un mensaje al servidor con el cambio de arma, entonces el servidor se encarga de enviar un mensaje al resto de clientes para que cambien el modelo de arma que tiene ese enemigo por el nuevo modelo.