.UBAfiuba
FACULTAD DE INGENIERÍA

Counter Strike 2D

Informe técnico

Taller de programación - 1° cuatrimestre 2025 - Cátedra: Veiga

Federico Honda - 110766

Martina Gualdi - 110513

Santiago Varga - 110561

Thiago Baez - 110703

.UBAfiuba
FACULTAD DE INGENIERÍA

Introducción

Este informe técnico describe el funcionamiento del programa a nivel de código. Su objetivo es proporcionar a futuros desarrolladores una comprensión profunda del sistema para que puedan mejorar el proyecto o adaptarlo según sus necesidades. Asimismo, está pensado para aquellos interesados en conocer el funcionamiento interno del juego.

Documentación

Servidor

El servidor es responsable de:

- Controlar la lógica del juego.
- Comunicar el estado del juego a todos los jugadores conectados.
- Administrar la sincronización y la coexistencia de los jugadores.

Cliente

El cliente es el encargado de:

- Representar visualmente el estado del juego para el jugador.
- Enviar las actualizaciones del jugador al servidor.
- Gestionar y utilizar las texturas y sonidos requeridos.

Editor

El editor es el encargado de:

- Permitir la creación y edición de niveles para ser utilizados en el juego.
- Gestionar un sistema de capas para evitar superposición no deseada de elementos.

Valgrind

Para correr el server con valgrind se debe anteponer al comando de ejecución “valgrind”:

valgrind ./server configuracion.yaml

Al jugar una partida y cerrar el servidor se obtiene la siguiente salida de valgrind:

```
==27823==
==27823== HEAP SUMMARY:
==27823==     in use at exit: 0 bytes in 0 blocks
==27823==   total heap usage: 1,277,737 allocs, 1,277,737 frees, 90,714,323 bytes allocated
==27823==
==27823== All heap blocks were freed -- no leaks are possible
==27823==
==27823== For lists of detected and suppressed errors, rerun with: -s
```

.UBAfiuba
FACULTAD DE INGENIERÍA

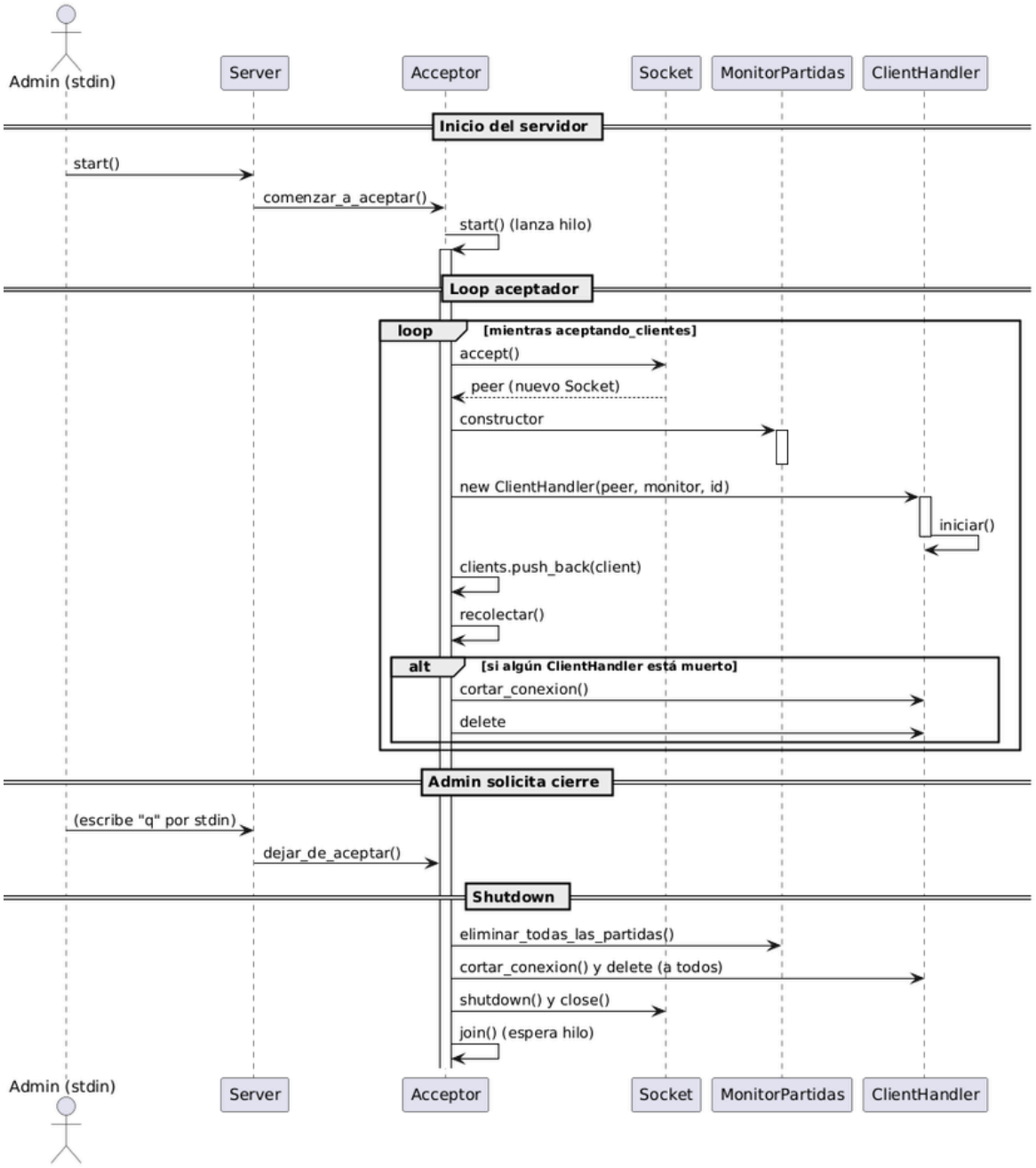
Organización de directorios

- **client_src/** : Directorio que contiene todo el código fuente de todo lo relacionado al cliente, se produce la comunicación con el servidor y todo lo gráfico en general. Se encarga de conectarse a un servidor, enviar peticiones en forma de comandos y traducir la información que este le envía a un formato gráfico amigable.
- **common_src/:** Directorio donde se encuentran clases y estructuras básicas que comparten tanto el servidor como el cliente.
- **server_src/:** Directorio que contiene todo el código fuente de toda la lógica del juego. El servidor se encarga de aceptar conexiones entrantes, añadir el usuario a una partida y procesar todos los comandos que este último le envía para posteriormente responderle con el estado del juego (snapshot).
- **editor/ :** contiene el código fuente del editor de niveles.
- **gfx/:** contiene todos los spritesheets que se usan para dibujar el cliente.
- **sfx/:** contiene todos archivos de los sonido (chunks) que se usan durante el juego, en formato .ogg y .wav
- **tests/ :** contiene el código fuente de los tests que se realizan para probar la conexión del socket.
- **Documentacion/ :** en este directorio encontrarás todos los PDFs con la información útil tanto para el usuario, como esta guía técnica.

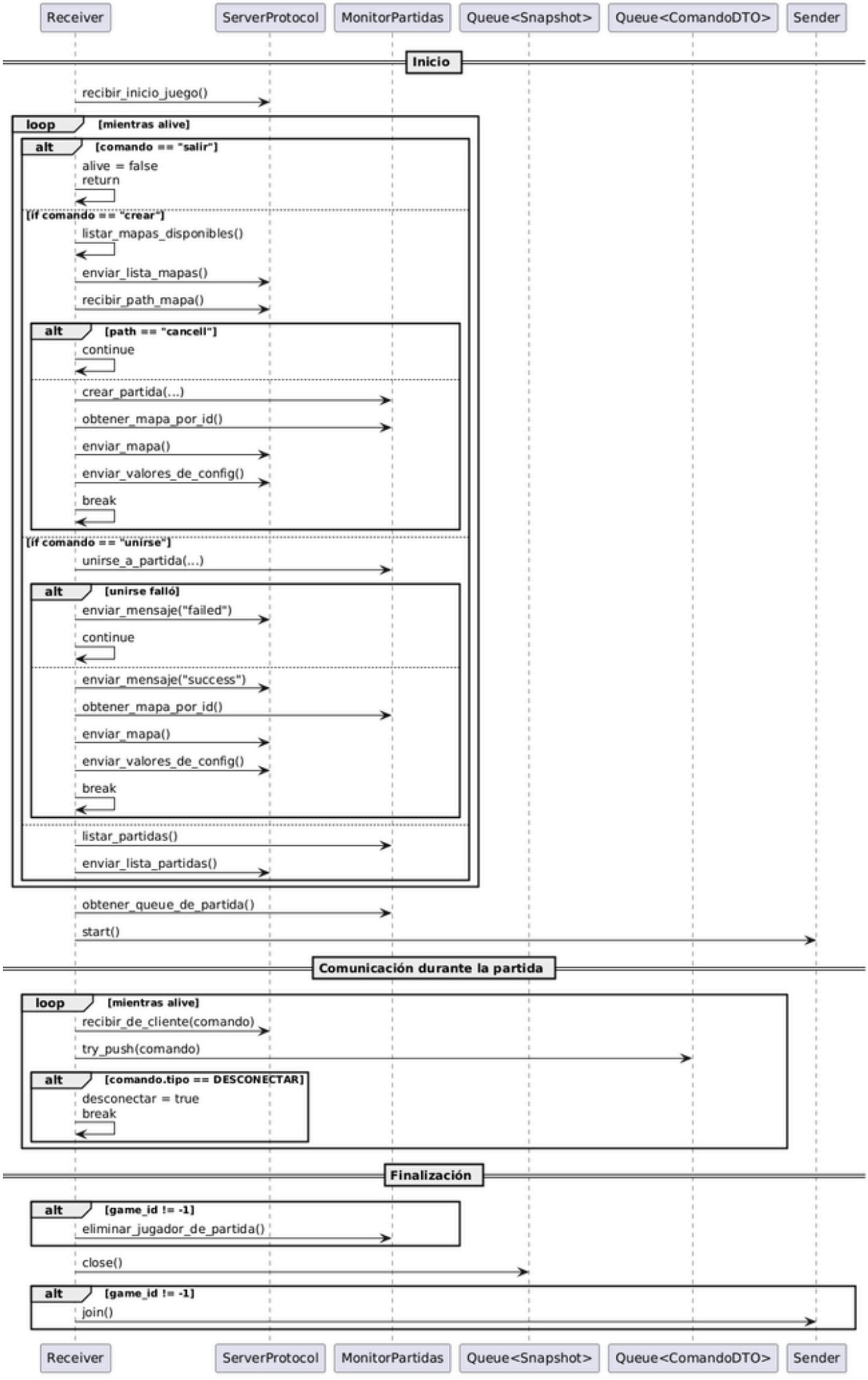
.UBAfiuba
FACULTAD DE INGENIERÍA

Diagramas de clase y secuencia

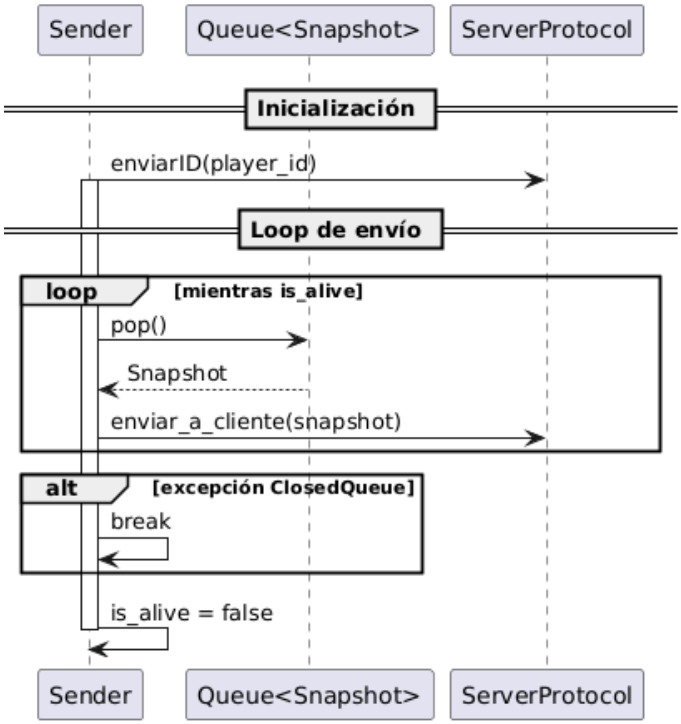
Server



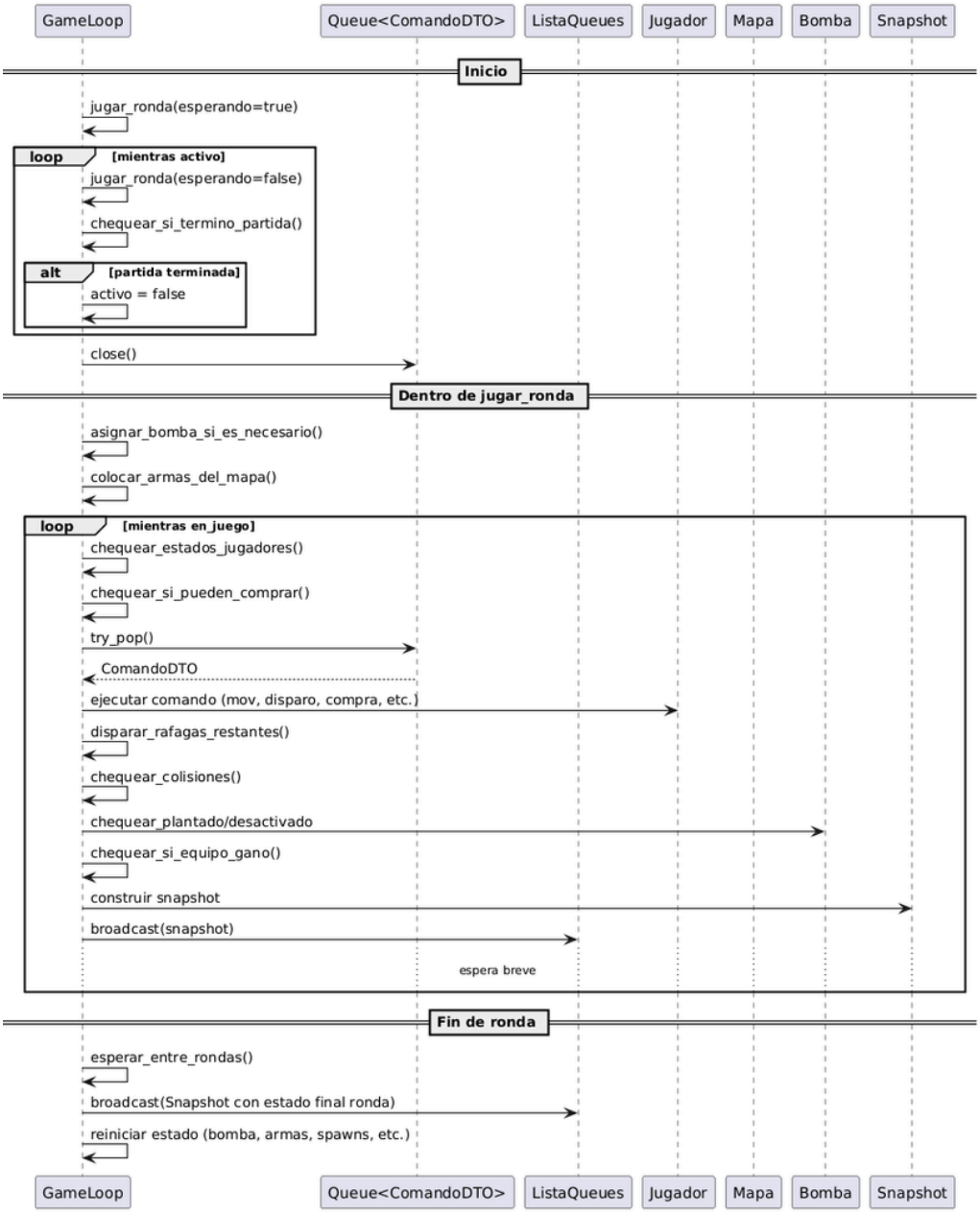
Hilo aceptador.



Hilo receiver.


Sender
Queue<Snapshot>
ServerProtocol
Inicialización
enviarID(player_id)
Loop de envío
loop
[mientras is_alive]
pop()
Snapshot
enviar_a_cliente(snapshot)
alt
[excepción ClosedQueue]
break
is_alive = false
Sender
Queue<Snapshot>
ServerProtocol

Hilo sender.



Hilo gameloop.

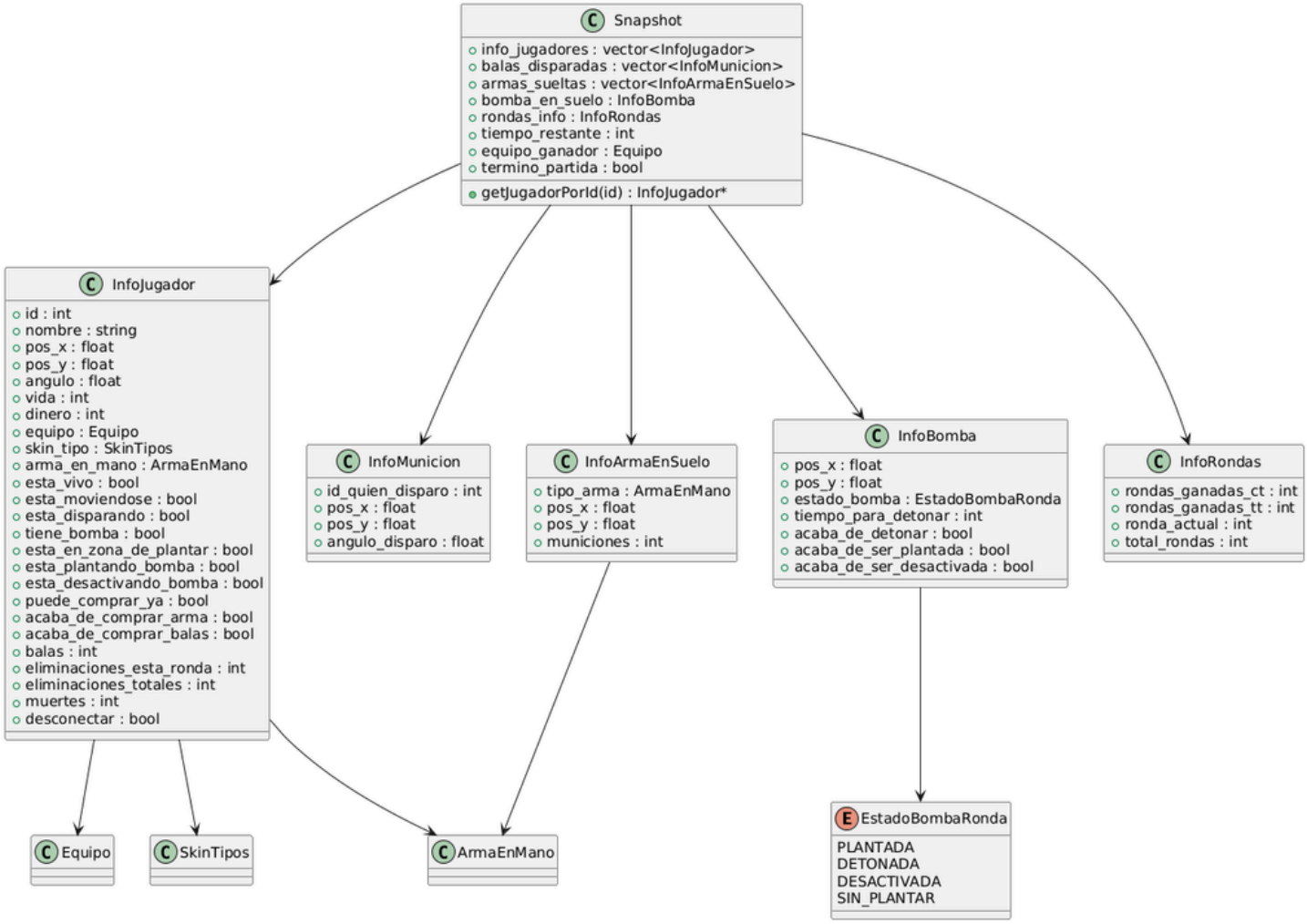


Diagrama de clase: arma.

Common

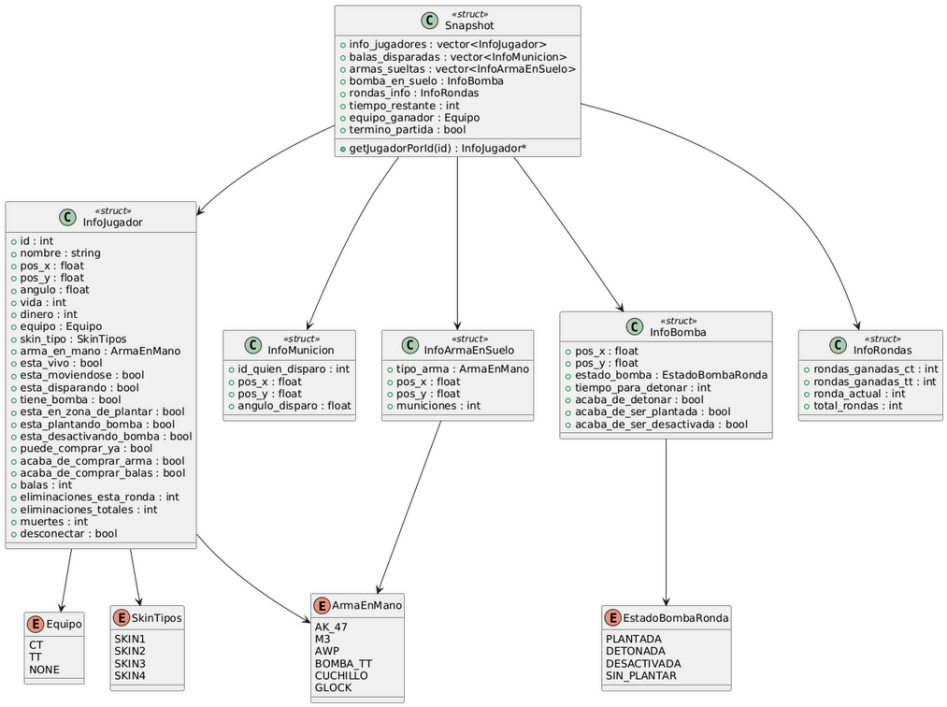


Diagrama de clase: snapshot.

Cliente

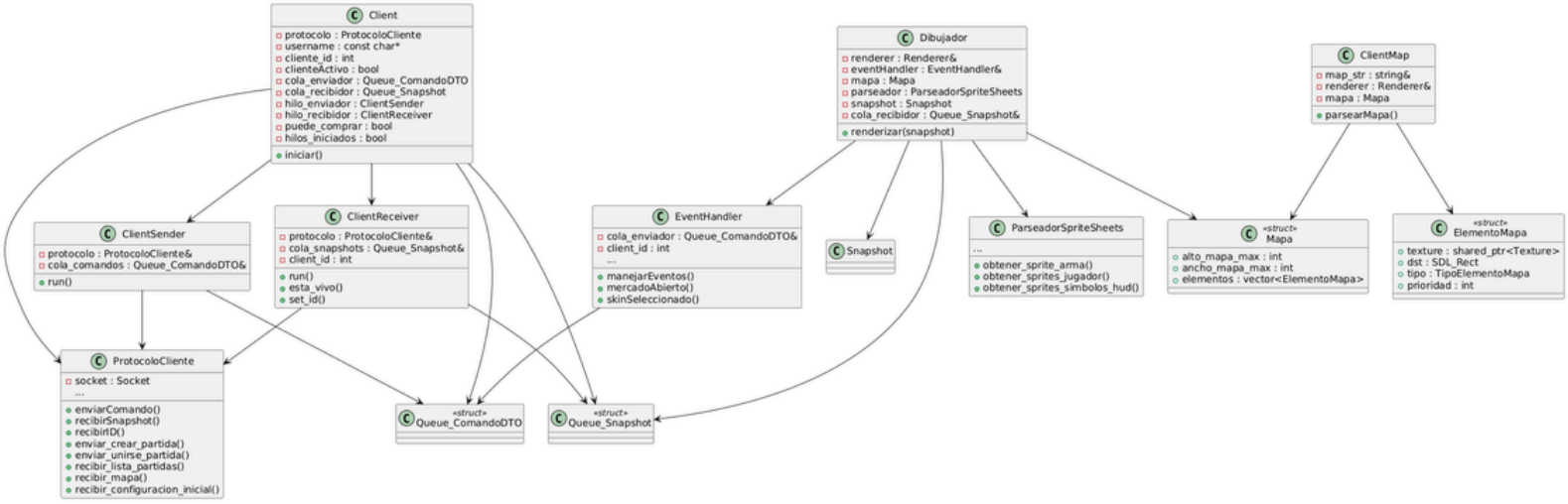
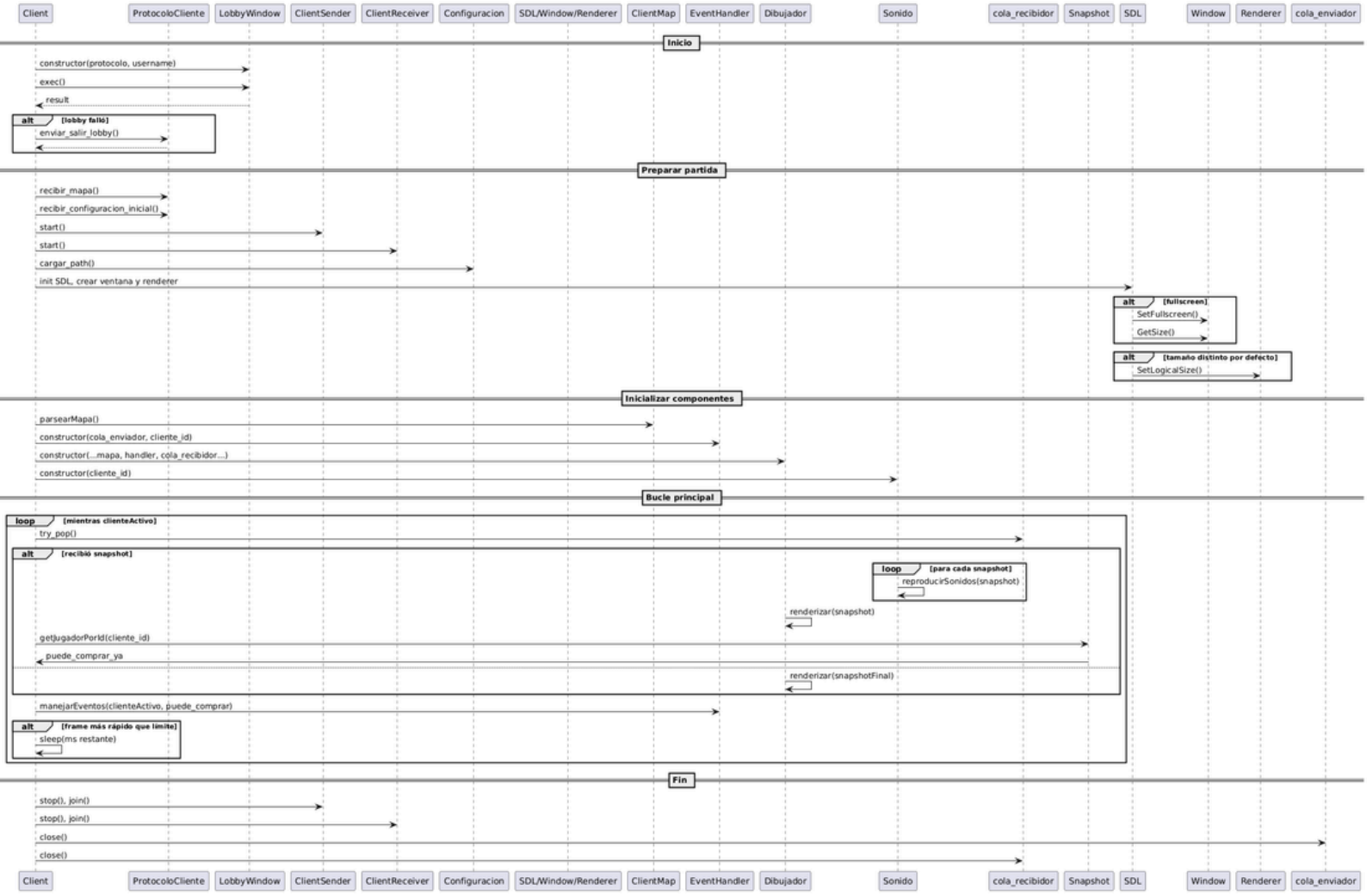
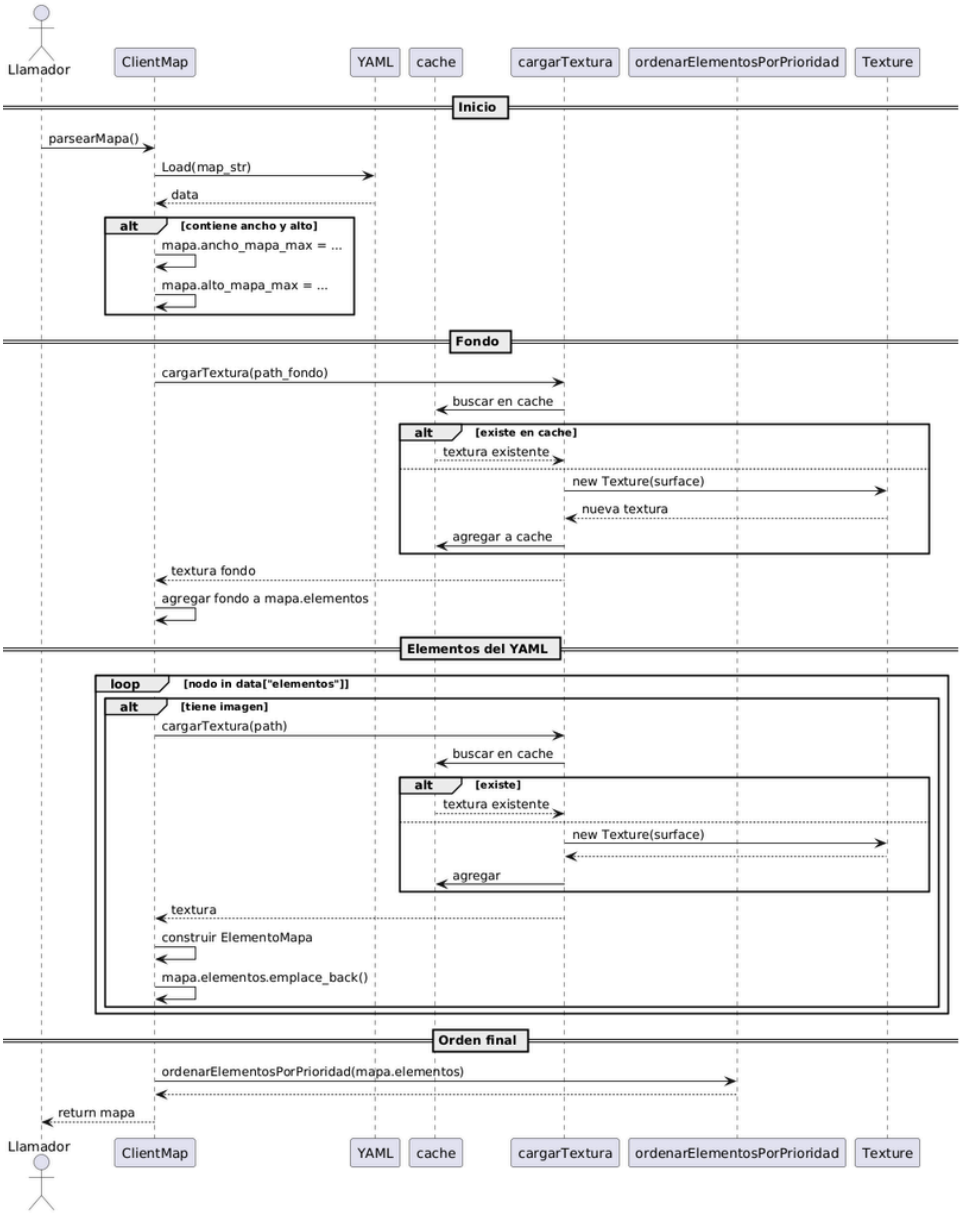


Diagrama de clase: Cliente



Ejecución cliente.



Ejecución de la carga del mapa desde un YAML.

Editor

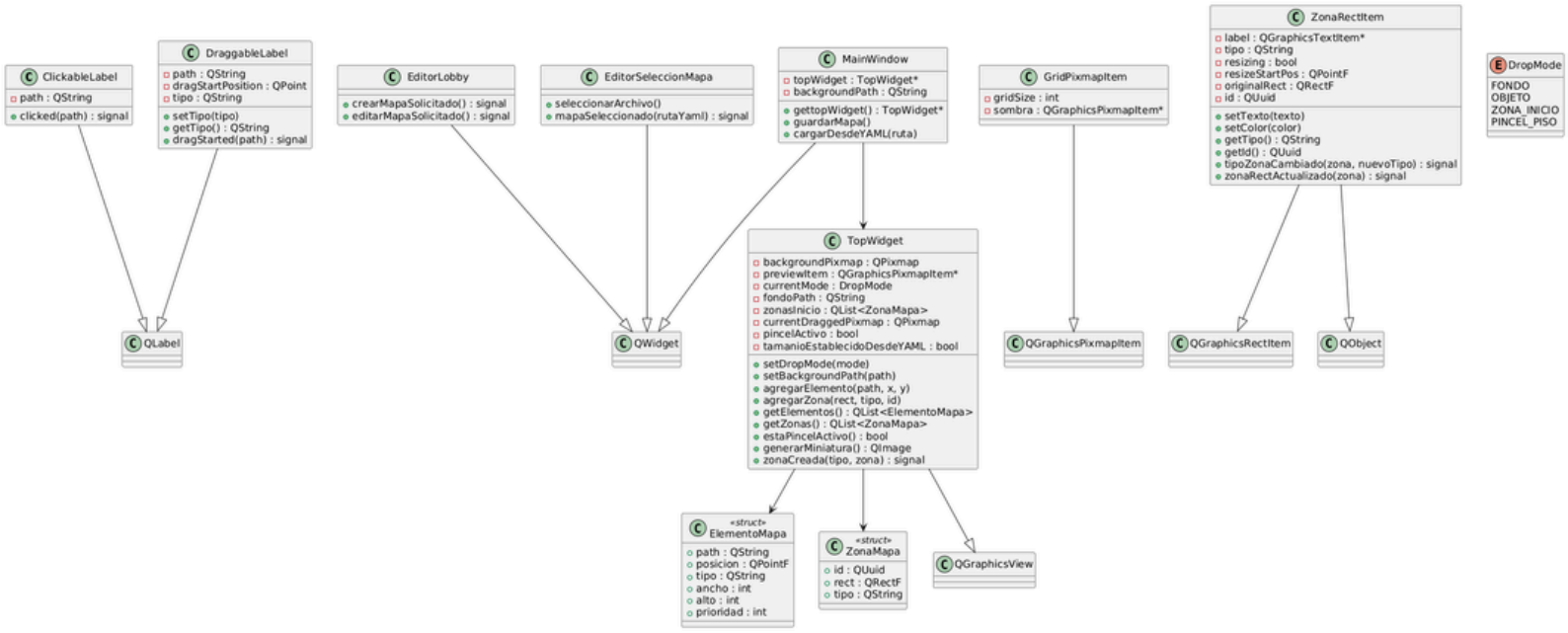


Diagrama de clases: editor.