



ESCUELA SUPERIOR DE INFORMÁTICA
UNIVERSIDAD DE CASTILLA – LA MANCHA

Práctica 1

Tres en raya definitivo

Javier Marchán Loro
Ángel Peralta López
Rubén Pérez Pascual
Antonio Ruedas García

Asignatura: Ingeniería del Software II
Titulación: Ingeniería Informática
Fecha: 11 de febrero de 2014

1. Introducción

Este documento describe el análisis, desarrollo e implementación llevados a cabo para obtener el juego **Tres en raya definitivo**¹.

Se trata de una aplicación cliente-servidor para escritorio que se comunica a través de RMI. El lenguaje utilizado para la implementación ha sido Java.

Este documento pretende que el lector obtenga una idea clara de la composición del sistema, para ello se ha dividido en distintas secciones. En primer lugar los requisitos funcionales ayudan a entender el objetivo buscado. La arquitectura del sistema junto con los casos de uso son las secciones más importantes, ya que ayudarán al lector a hacerse una imagen mental, a alto nivel, del sistema desarrollado.

El resto de secciones profundizan en el conocimiento del sistema.

¹Este documento no explica las reglas del juego

2. Requisitos funcionales

2.1. Registrarse

Caso de uso	Registrarse
Precondición Escenario general	No está registrado 1. El usuario pulsa el botón de registrarse en la ventana de inicio. 2. Se muestra la ventana de registro. 3. El usuario introduce su email y su contraseña en la ventana de registro. 4. El usuario pulsa el botón de confirmación. 5. Se espera confirmación por parte del servidor. 6. La ventana de inicio informa al usuario del éxito de la operación.
Poscondición	Un nuevo usuario existe en el sistema.
Precondición Escenario alternativo	Ya está registrado 1. El usuario pulsa el botón de registrarse en la ventana de inicio. 2. Se muestra la ventana de registro. 3. El usuario introduce su email y contraseña en la ventana de registro. 4. El usuario pulsa el botón de confirmación. 5. Se espera confirmación por parte del servidor. 6. La ventana de inicio informa al usuario que ya se encuentra registrado.
Poscondición	No se registra el usuario de nuevo.

2 Requisitos funcionales

2.2. Iniciar sesión

Caso de uso	Iniciar sesión
Precondición Escenario general	Usuario registrado y aún no logeado 1. El usuario introduce su email y contraseña en la ventana de inicio. 2. El usuario pulsa el botón de “Iniciar sesión”. 3. Se espera validación por parte del servidor. 4. Se cierra la ventana de login y se abre la lista de jugadores.
Poscondición	Se ejecuta el caso de uso “Ver lista de jugadores”
Precondición Escenario alternativo	Datos de acceso incorrectos 1. El usuario introduce su email y contraseña en la ventana de inicio. 2. El usuario pulsa el botón de “Iniciar sesión”. 3. Se espera validación por parte del servidor. 4. Se informa al usuario que los datos de acceso son incorrectos.
Poscondición	Usuario no logeado.
Precondición Escenario alternativo	Usuario ya logeado. 1. El usuario introduce su email y contraseña en la ventana de inicio. 2. El usuario pulsa el botón de “Iniciar sesión”. 3. Se espera validación por parte del servidor. 4. Se informa al usuario que ya está logeado.
Poscondición	Usuario no logeado de nuevo.

2 Requisitos funcionales

2.3. Cerrar sesión

Caso de uso	Cerrar sesión
Precondición Escenario general	Sesión iniciada <ol style="list-style-type: none">1. El usuario pulsa el botón de cerrar el programa.2. Se manda una petición de cierre de sesión al servidor.3. El servidor cierra la sesión del usuario.4. Se cierra el programa.
Poscondición	El usuario deja de estar activo en el servidor y deja de aparecer en la lista de jugadores.
Precondición Escenario alternativo	Sesión iniciada y jugador en partida <ol style="list-style-type: none">1. El usuario pulsa el botón de cerrar el programa.2. Se manda una petición de cierre de sesión al servidor.3. El servidor elimina la partida y cierra la sesión del usuario.4. Se cierra el programa.
Poscondición	El usuario deja de estar activo en el servidor y deja de aparecer en la lista de jugadores.

2.4. Salir de la partida

Caso de uso	Salir de la partida
Precondición Escenario general	Estar conectado a una partida. <ol style="list-style-type: none">1. El usuario pulsa el botón cerrar partida en la ventana de juego.2. Se comunica al servidor que elimmine la partida.3. Se cierra la ventana de juego.
Poscondición	Se sale de la partida y aparece la ventana principal.

2 Requisitos funcionales

2.5. Retar jugador

Caso de uso	Retar jugador
Precondición Escenario general	Sesión iniciada y oponente en el sistema <ol style="list-style-type: none">1. El usuario selecciona un jugador de la lista y pulsa "Retar".2. Se manda la petición al servidor que a su vez se lo comunica al oponente.3. El servidor busca al oponente y le envia la solicitud de reto.
Poscondición	Usuario espera confirmación del oponente.

2.6. Aceptar reto

Caso de uso	Aceptar reto
Precondición Escenario general	Sesión iniciada y petición de reto realizada <ol style="list-style-type: none">1. Se abre ventana emergente de invitación a partida.2. El usuario pulsa el botón de aceptar.3. Se notifica al servidor de la aceptación.
Poscondición	Se crea una partida.
Precondición Escenario alternativo	Sesión iniciada y petición de reto realizada <ol style="list-style-type: none">1. Se abre ventana emergente de invitación a partida.2. El usuario pulsa el botón de cancelar.3. Se notifica al servidor de la cancelación.
Poscondición	No se crea partida y se vuelve a la GUI de la lista de jugadores.

2 Requisitos funcionales

2.7. Realizar movimiento

Caso de uso	Realizar movimiento válido
Precondición Escenario general	Existencia de una partida 1. El usuario coloca una pieza en el tablero de juego. 2. Se comprueba la validez del movimiento. 3. Se manda información del movimiento al servidor.
Poscondición	Tablero modificado.
Precondición Escenario alternativo	Realizar movimiento no válido 1. El usuario coloca una pieza en el tablero de juego. 2. Se comprueba la validez del movimiento. 3. Se informa al usuario de que el movimiento no es válido.
Poscondición	Tablero no modificado.

2.8. Ver la lista de jugadores

Caso de uso	Ver la lista de jugadores
Precondición Escenario general	Iniciar sesión 1. Se abre la ventana de la lista de jugadores y recibe la lista de usuarios conectados del servidor.
Poscondición	La lista de jugadores está disponible para el usuario.
Precondición Escenario alternativo	Jugador ya iniciado sesión y otro jugador se conecta. 1. El servidor envía al usuario la lista de jugadores actualizada.
Poscondición	Ambos jugadores con la lista de jugadores actualizada.

2 Requisitos funcionales

2.9. Enviar Petición de Reto

Caso de uso	Enviar petición de Reto
Precondición Escenario general	Usuarios logeados en el sistema y petición de reto realizada. <ol style="list-style-type: none">Al servidor le llega una petición de reto por parte de un usuario.El servidor envía al otro jugador la petición de reto.
Poscondición	

2.10. Registrar un movimiento

Caso de uso	Registrar un movimiento
Precondición	Iniciar sesión, Unirse a una partida y Conectarse a una partida
Escenario general	<ol style="list-style-type: none">Al servidor le llega el movimiento realizado por un usuario.El servidor almacena el movimiento en la base de datos.El servidor notifica al otro usuario que ya puede realizar su movimiento.
Poscondición	Se realiza el caso de uso: "Enviar actualización de la partida".

2 Requisitos funcionales

2.11. Responder a solicitud de reto

Caso de uso	Responder a solicitud de reto
Precondición Escenario general	Iniciar sesión 1. El servidor envía al usuario la solicitud de reto del otro usuario. 2. El cliente recibe la solicitud y la envía al controlador. 3. El controlador la envía a la ventana de jugadores para que muestre que el jugador ha sido retado 4. El usuario elige la acción en la ventana (si acepta o si no). 5. La ventana de juego envía la información al controlador. 6. La fachada envía la respuesta al servidor por medio del proxy.
Poscondición	Respuesta a reto enviada.

2.12. Realizar movimiento

Caso de uso	Realizar Movimiento
Precondición Escenario general	El usuario debe haberse unido a una partida y estar conectado a ella. 1. El usuario pulsa sobre la casilla del tablero a colocar. 2. La interfaz avisa al controlador del movimiento. 3. El controlador envía el movimiento a la fachada. 4. La fachada envía el movimiento a Tablero9x9. 5. Tablero9x9 comprueba la legalidad del movimiento. 6. Si el movimiento es válido se lo envía al proxy y avisa al controlador para realizar el camino inverso hacia la interfaz y colocar la ficha. 7. Si el movimiento es inválido se lanza una excepción que se recoge en el controlador. 8. El controlador envía la excepción a la interfaz de usuario para comunicarle el error que ha cometido.
Postcondición	Movimiento realizado.

3 Otros requisitos

3. Otros requisitos

3.1. Requisitos de la interfaz de usuario

3.1.1. Ventana de inicio

Descripción: Da la bienvenida al usuario.

Debe permitir al usuario ejecutar el caso de uso “Iniciar sesión” e introducir los siguientes datos:

Dato	Tipo	Descripción
NombreUsuario	Texto	Nombre del usuario
Contraseña	Texto	Contraseña en el sistema

3.1.2. Ventana de registro

Descripción: Permite registrarse al usuario.

Es necesario que permita al usuario introducir los siguientes datos:

Dato	Tipo	Descripción
Nombre	Texto	Nombre del usuario
eMail	Texto	Dirección de correo electrónico
Contraseña	Texto	Contraseña en el sistema
Conf. Contraseña	Texto	Confirmación de contraseña

3.1.3. Ventana principal

Descripción: Mostrará al usuario las partidas disponibles y las partidas en juego.

Es necesario que permita al usuario ejecutar los siguientes casos de uso:

1. Crear una partida.
2. Unirse a una partida.
3. Conectarse a una partida.
4. Ver lista de partidas.

3.1.4. Ventana Crear partida

Descripción: Permite al usuario crear una nueva partida.

Es necesario que permita al usuario introducir los siguientes datos:

3 Otros requisitos

Dato	Tipo	Descripción
Nombre	Texto	Nombre de la partida
Días de juego	Fecha	Días en los que se jugará la partida
Hora de Inicio de Juego	Hora	Hora en la que empieza o continua la partida los días indicados
Hora de Fin de Juego	Hora	Hora en la que termina la partida los días indicados

3.1.5. Ventana de juego

Descripción: Muestra al usuario todos los detalles de una partida y permite jugar en ella. Deberá mostrar el tablero de juego con la asignación de territorios.

Es necesario que permita al usuario ejecutar los siguientes casos de uso:

1. Desconectarse de una partida.
2. Realizar un movimiento.
3. Responder acción enemiga.
4. Comprar refuerzos.
5. Enviar petición de alianza.
6. Responder a petición de alianza.
7. Romper alianza.

3 Arquitectura del sistema

En la figura 1 se muestra una visión de la arquitectura del sistema. Se trata de una arquitectura cliente-servidor a través de RMI. El cliente usa un proxy para comunicarse con el servidor y éste conoce un proxy de cada cliente con el que se comunica. Toda la lógica del juego reside en el cliente

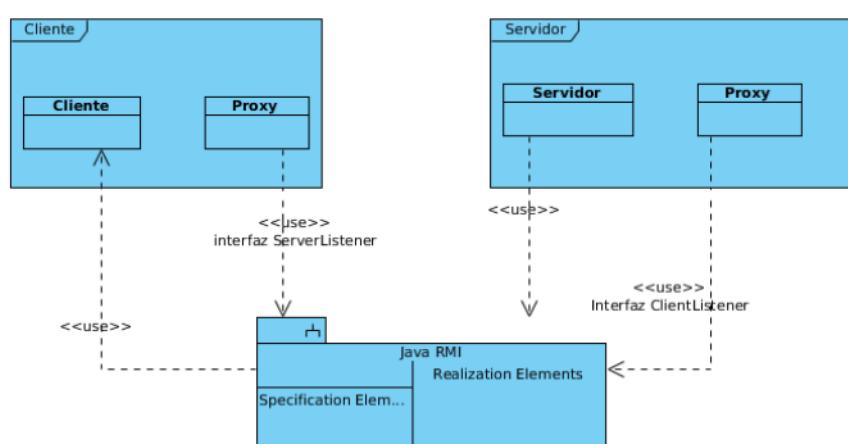


Figura 1: Arquitectura del sistema

3.1. Arquitectura del cliente

En la figura 2 se muestra la arquitectura del cliente. Se ha utilizado una arquitectura MVC. Además, se incluye una capa de presentación y una capa de comunicación:

- *Presentación:* Contiene las clases de las ventanas que se muestran al usuario y las interfaces que implementan.
- *Controlador:* Contiene la lógica de control. Se encarga de comunicar a la interfaz los cambios de estado de la capa de dominio y de modificar el estado del dominio atendiendo a los cambios en la interfaz del juego.
- *Dominio:* Contiene toda la lógica del juego. La clase Tablero9x9 es la clase principal e implementa las reglas del juego. Se utiliza el patrón fachada para obtener un punto de acceso único a la capa de dominio. La clase encargada de esta abstracción es la clase FTERD.
- *Comunicación:* Contiene las clases que realizan la función de *listener* como es el caso de la clase Cliente, y el proxy con el servidor. Estas

3 Arquitectura del sistema

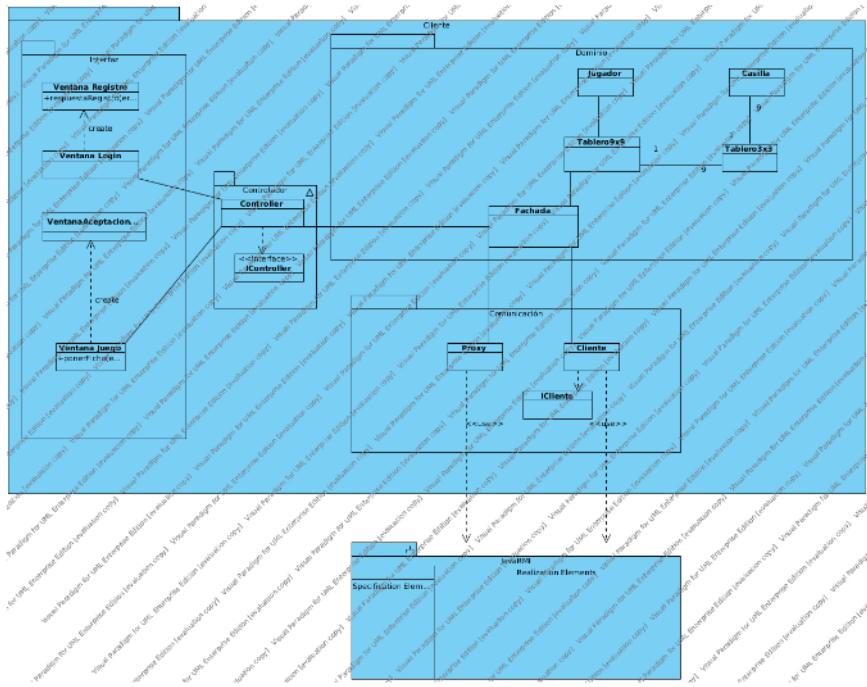


Figura 2: Arquitectura del cliente

clases solamente se encargan del envío de los respectivos mensajes al servidor o a la fachada del paquete de dominio.

3.2. Arquitectura del servidor

En la figura 3 se muestra la arquitectura multicapa del servidor:

- *Comunicación:* Contiene las clases encargadas de la comunicación, en este caso la clase servidor y su interfaz RMI. La clase servidor actúa como *listener* para escuchar solicitudes de los clientes. También responde a esos clientes ya que internamente guarda una hash con los clientes que se han dado de alta en el sistema.
- *Persistencia:* Contiene las clases encargadas de almacenar las partidas y sus respectivos tableros, jugadores y movimientos. Entre esas clases está la clase *Broker* que actúa como agente entre el subsistema de base de datos y el sistema del servidor. Se ha optado por utilizar un patrón de Fabricación Pura en lugar de seguir el patrón Experto para la

3 Arquitectura del sistema

persistencia. De esta forma, un conjunto de clases *DAO* especializadas se encargan de almacenar los diferentes aspectos del dominio.

- *Dominio*: Contiene las clases de dominio como las encargadas de representar a cualquier juego, además de la clase Fachada. Esta clase contiene una referencia a cada uno de los modelos que se están jugando en el sistema, es decir, contiene todos los tableros activos.

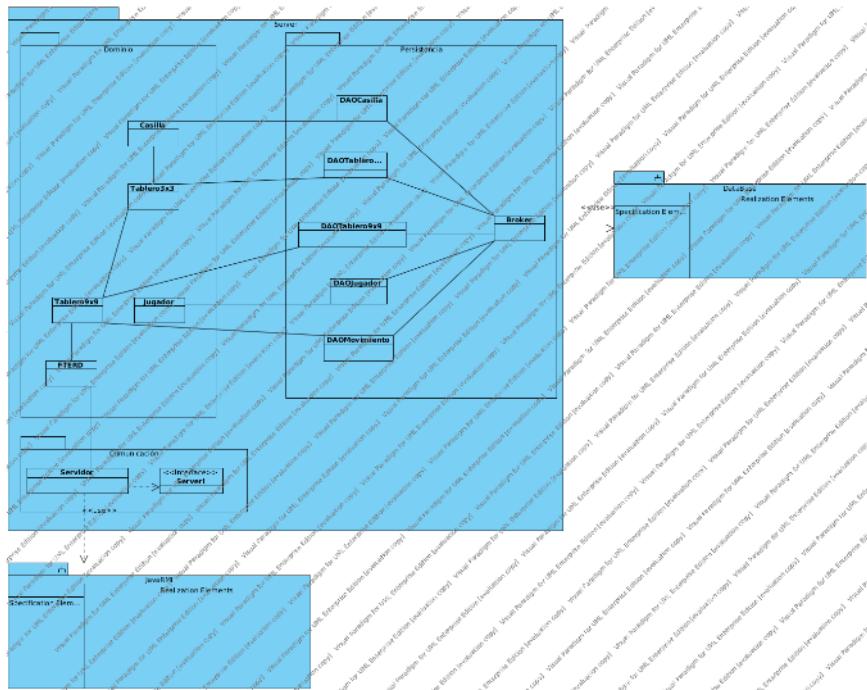
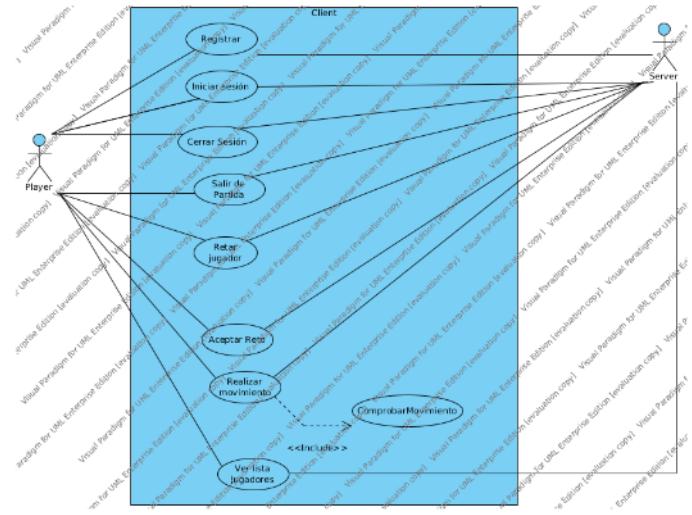


Figura 3: Arquitectura del servidor

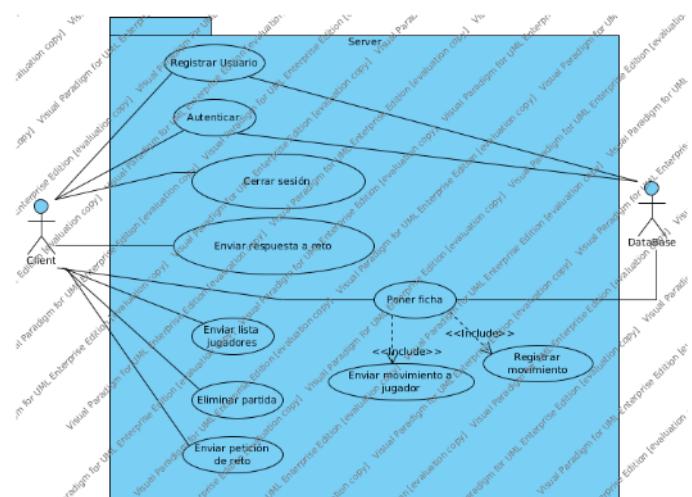
4 Casos de uso

4. Casos de uso

4.1. Diagrama de casos de uso del cliente



4.2. Diagrama de casos de uso del servidor

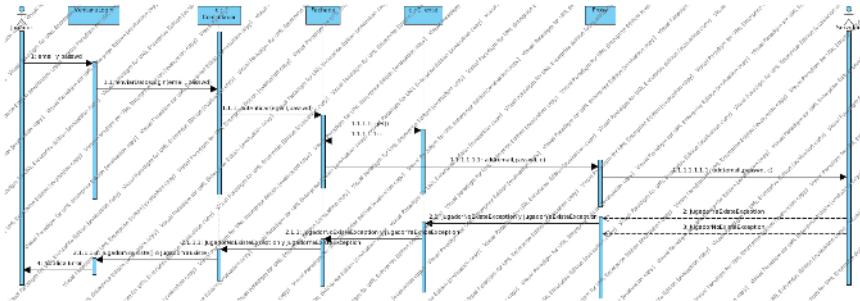


5 Diagramas de secuencia

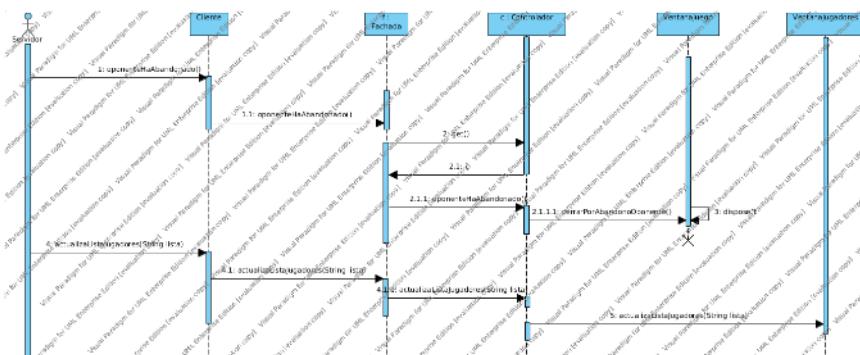
5. Diagramas de secuencia

5.1. Diagramas de secuencia del cliente

5.1.1. Autenticar



5.1.2. Cerrar sesión oponente

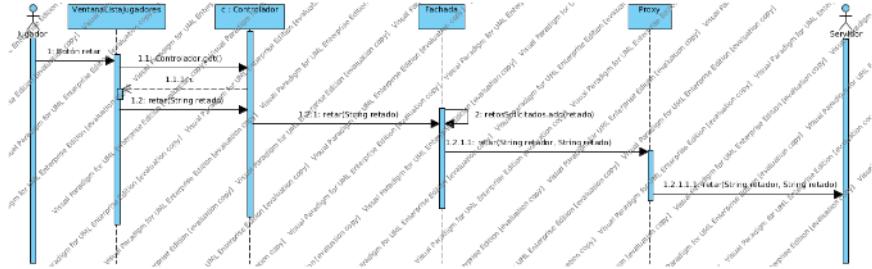


5.1.3. Enviar respuesta de reto

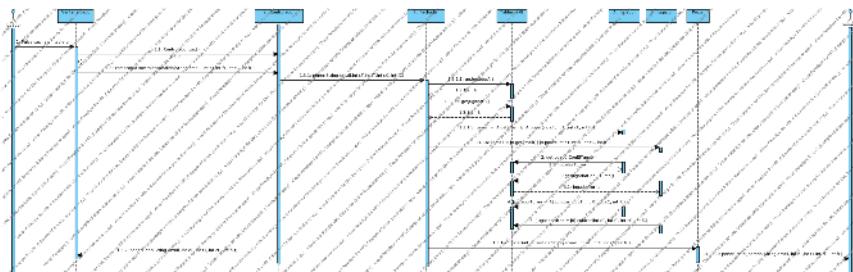


5 Diagramas de secuencia

5.1.4. Enviar reto



5.1.5. Realizar movimiento



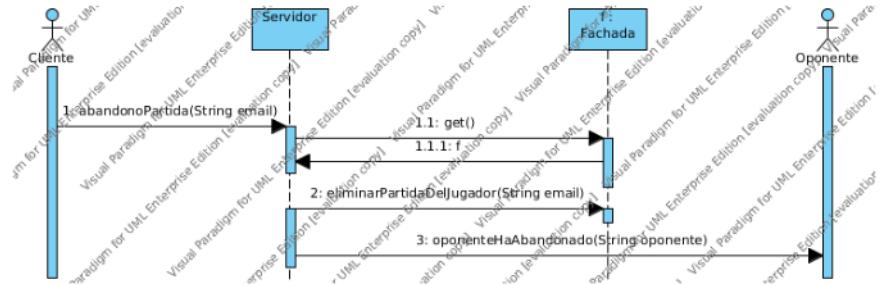
5.1.6. Recibir movimiento



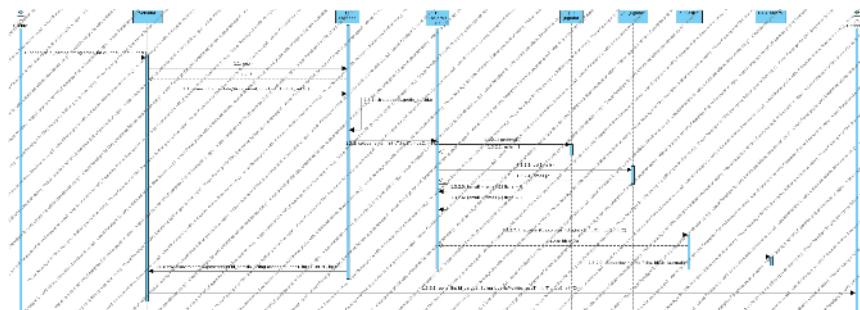
5 Diagramas de secuencia

5.2. Diagramas de secuencia del servidor

5.2.1. Cerrar sesión

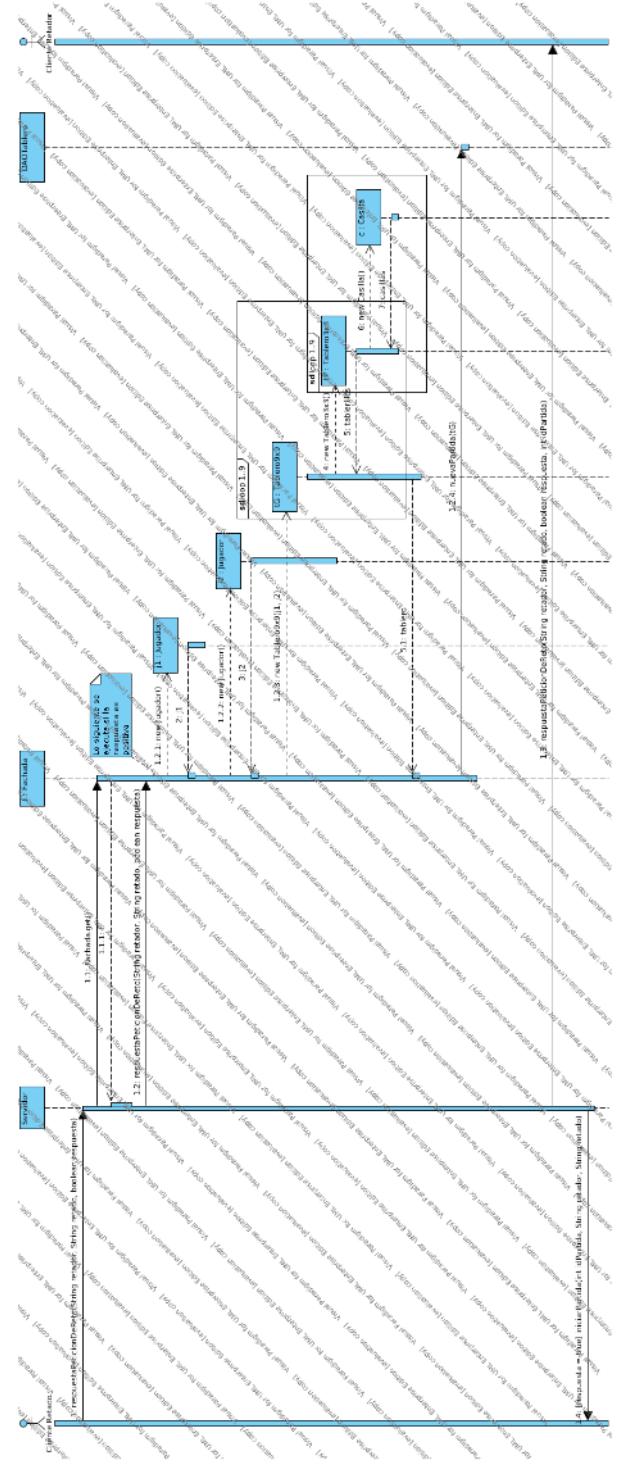


5.2.2. Realizar movimiento



5 Diagramas de secuencia

5.2.3. Enviar respuesta de reto

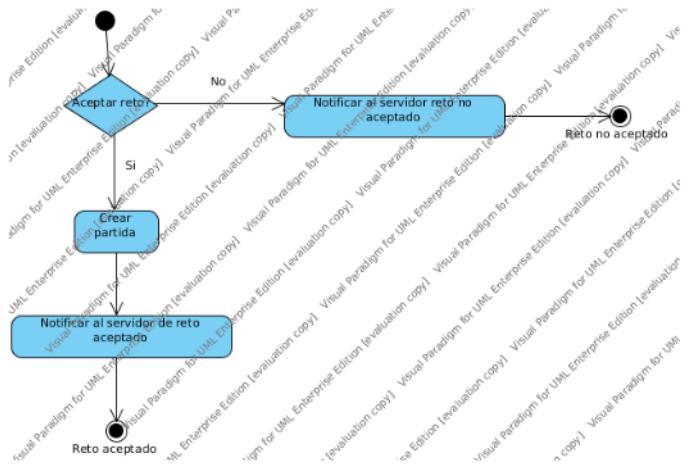


6 Máquinas de estado

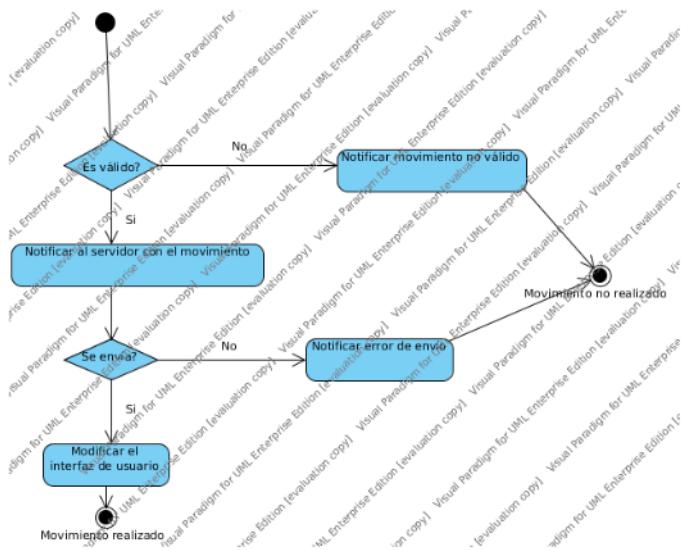
6. Máquinas de estado

6.1. Máquinas de estado del cliente

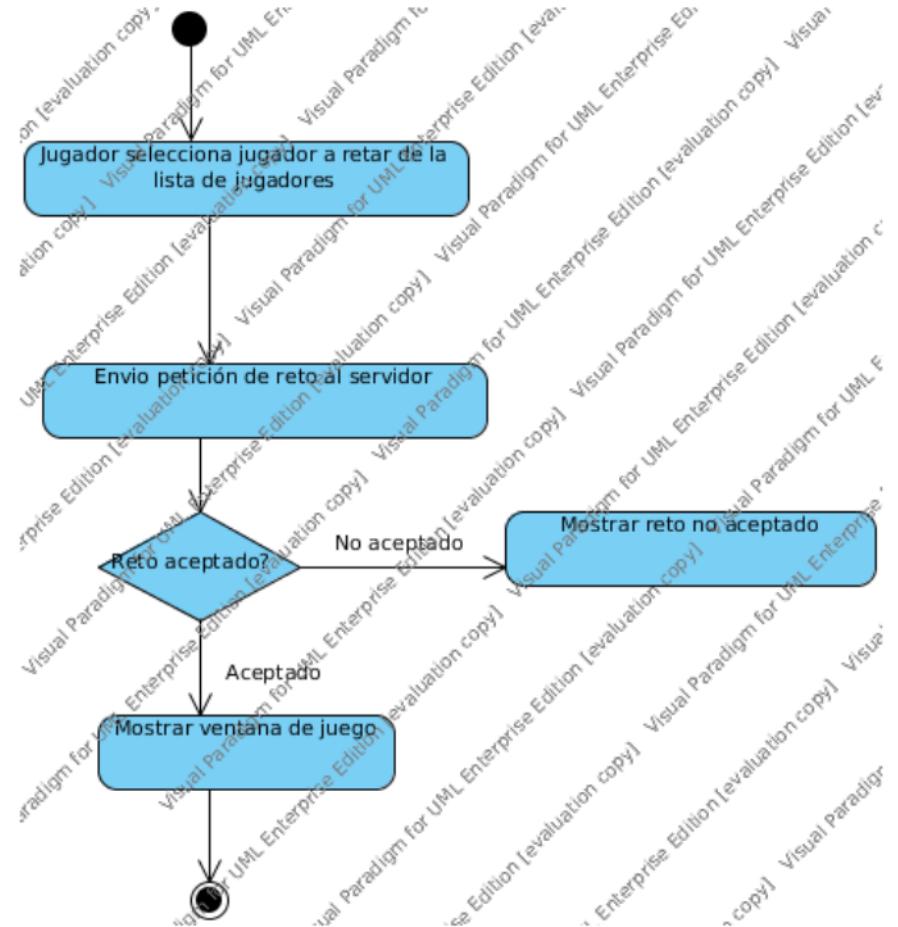
6.1.1. Aceptar reto



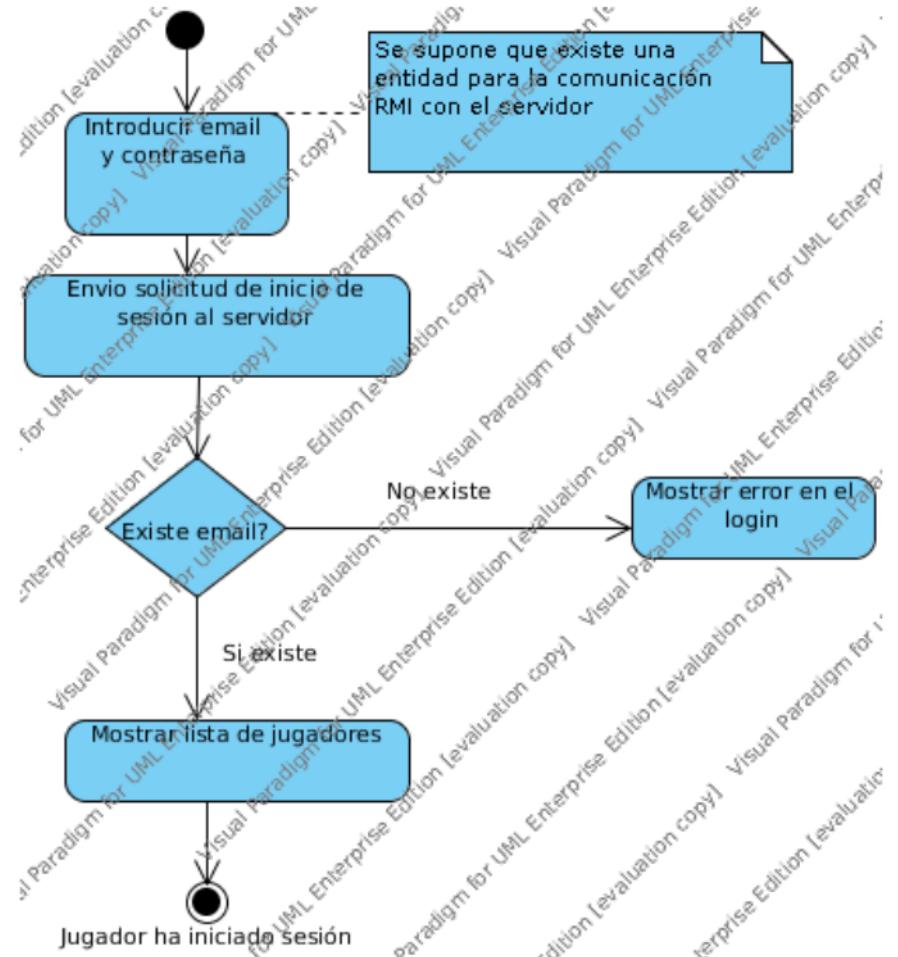
6.1.2. Realizar movimiento



6.1.3. Retar jugador

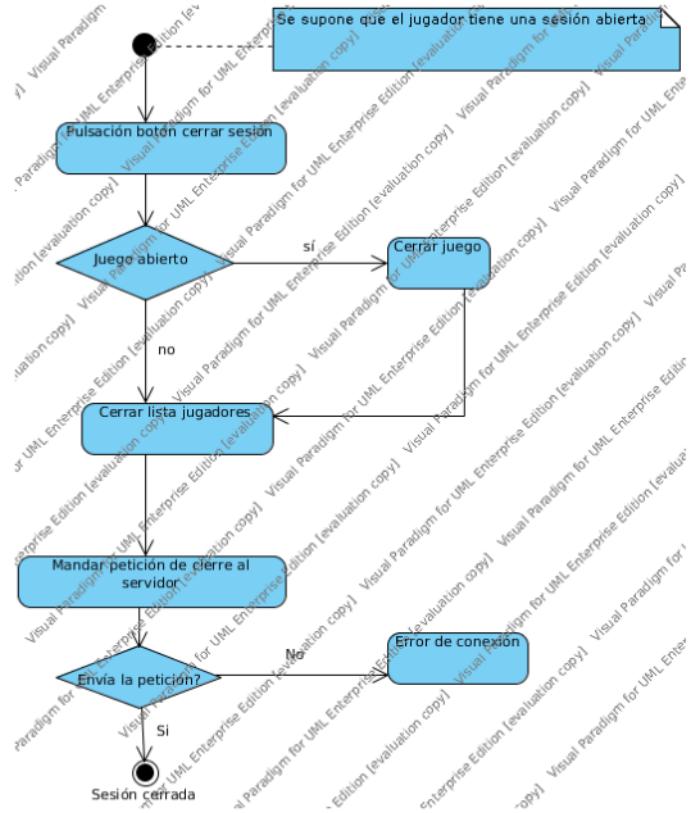


6.1.4. Iniciar sesión



6 Máquinas de estado

6.1.5. Cerrar sesión



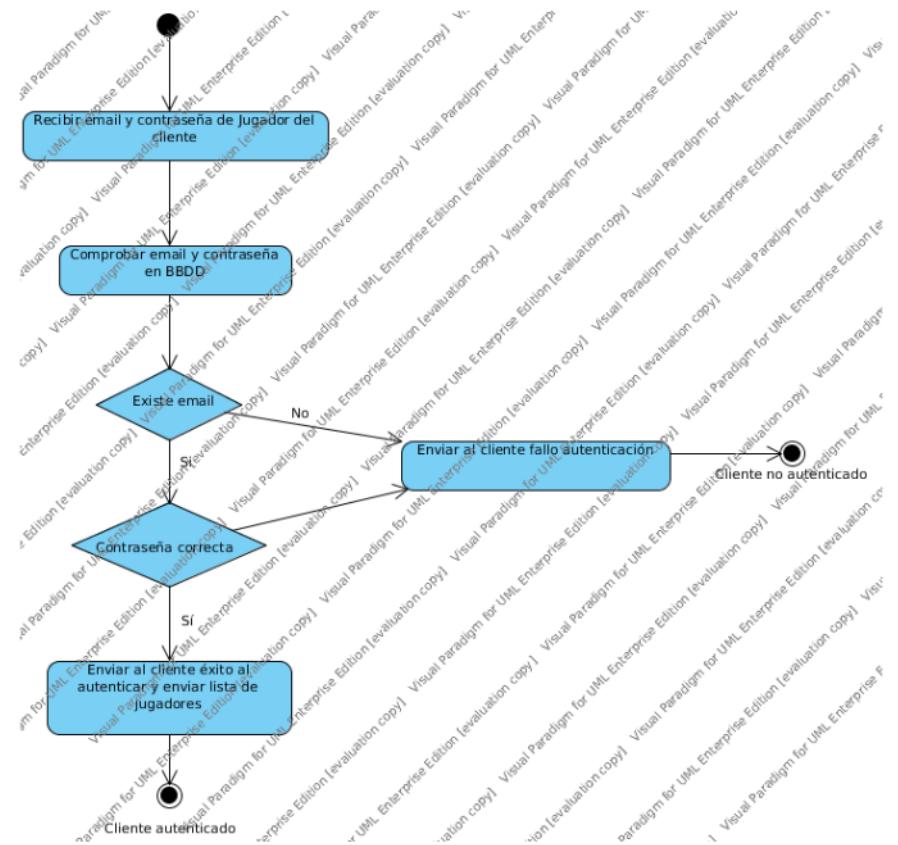
6.1.6. Salir de partida



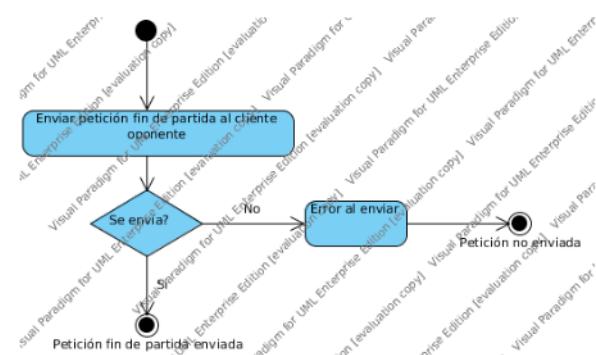
6 Máquinas de estado

6.2. Máquinas de estado del Servidor

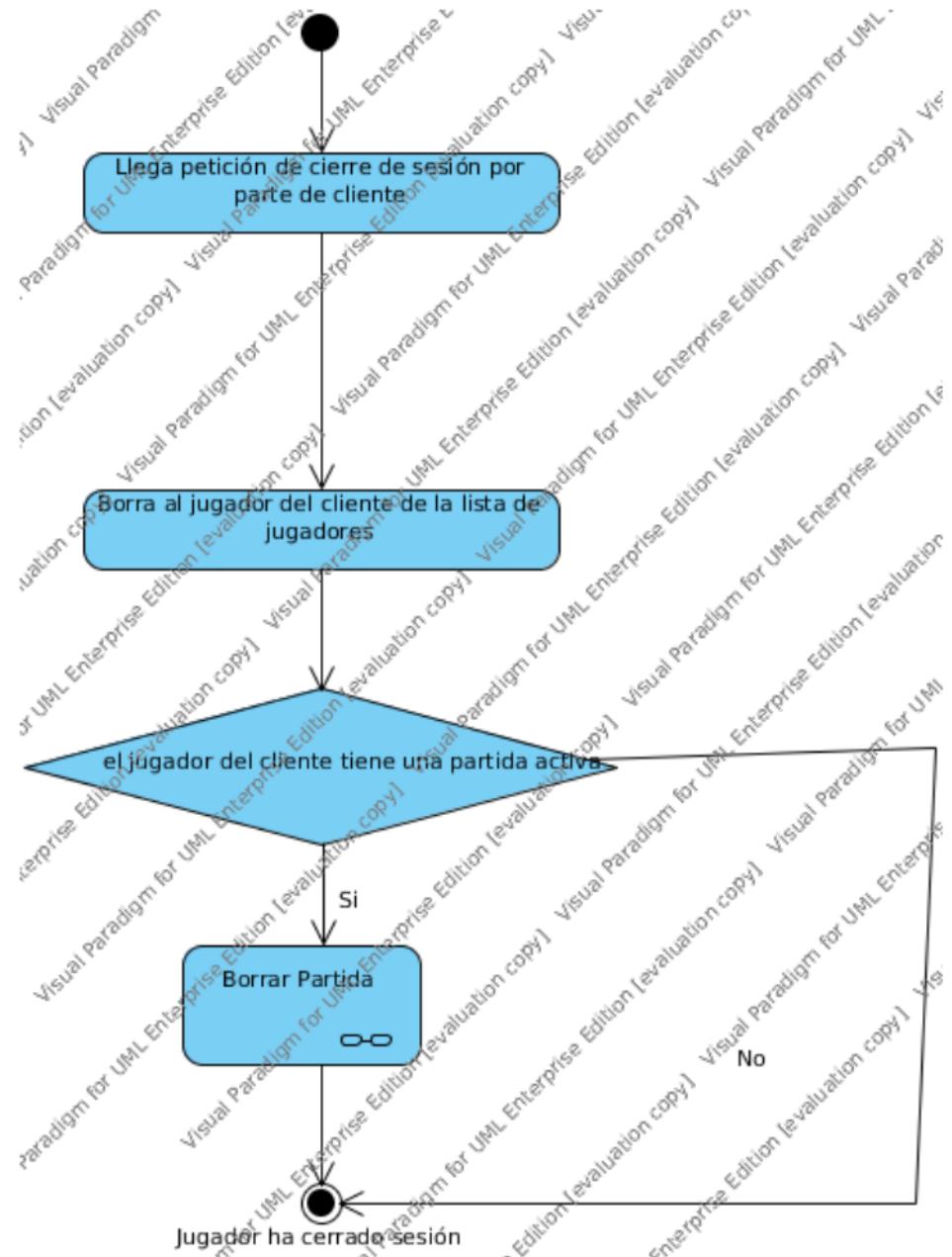
6.2.1. Autenticar



6.2.2. Borrar partida



6.2.3. Cerrar sesión

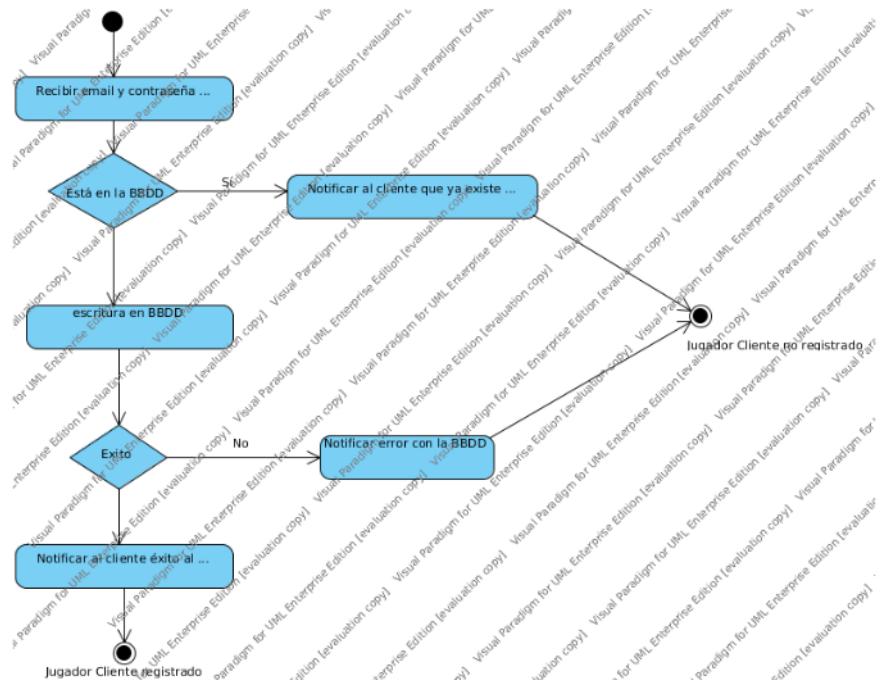


6 Máquinas de estado

6.2.4. Poner ficha

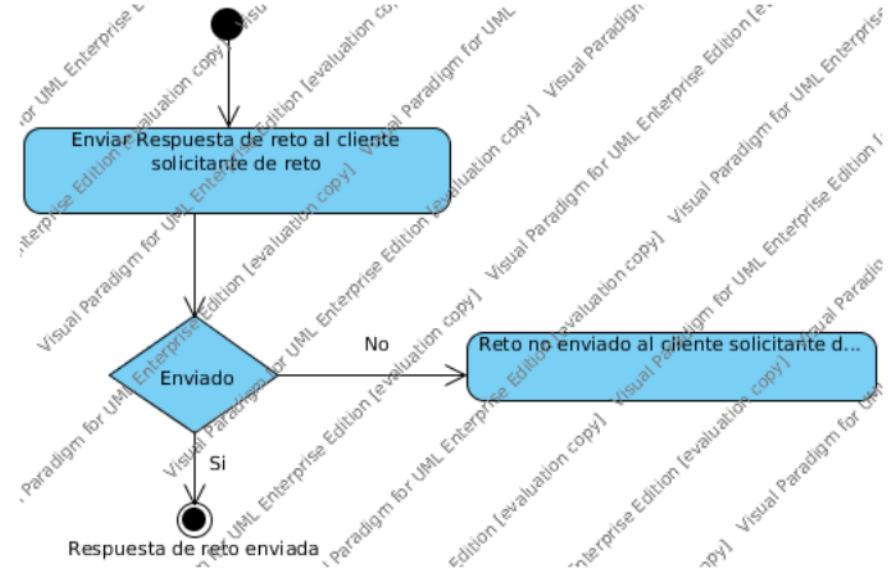


6.2.5. Registrar usuario

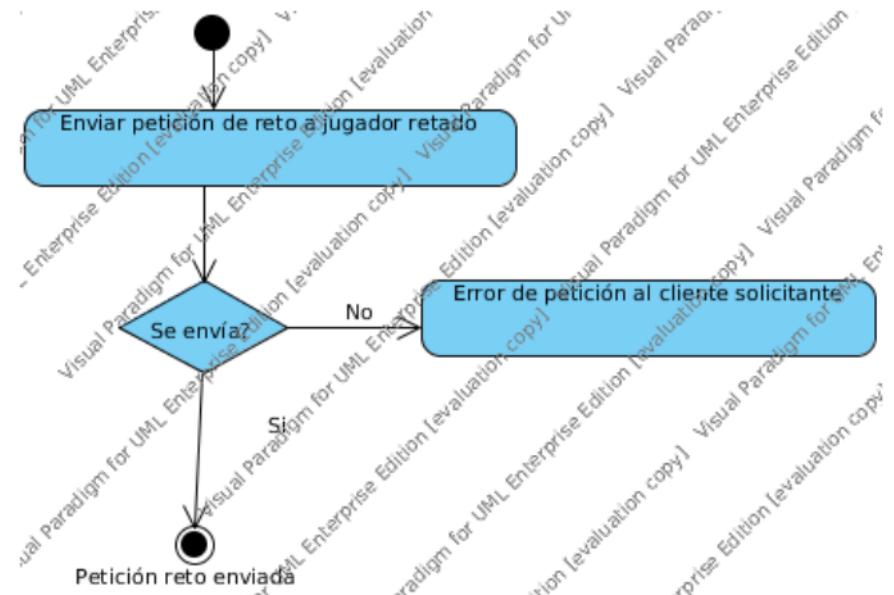


6 Máquinas de estado

6.2.6. Respuesta de reto



6.2.7. Retar jugador

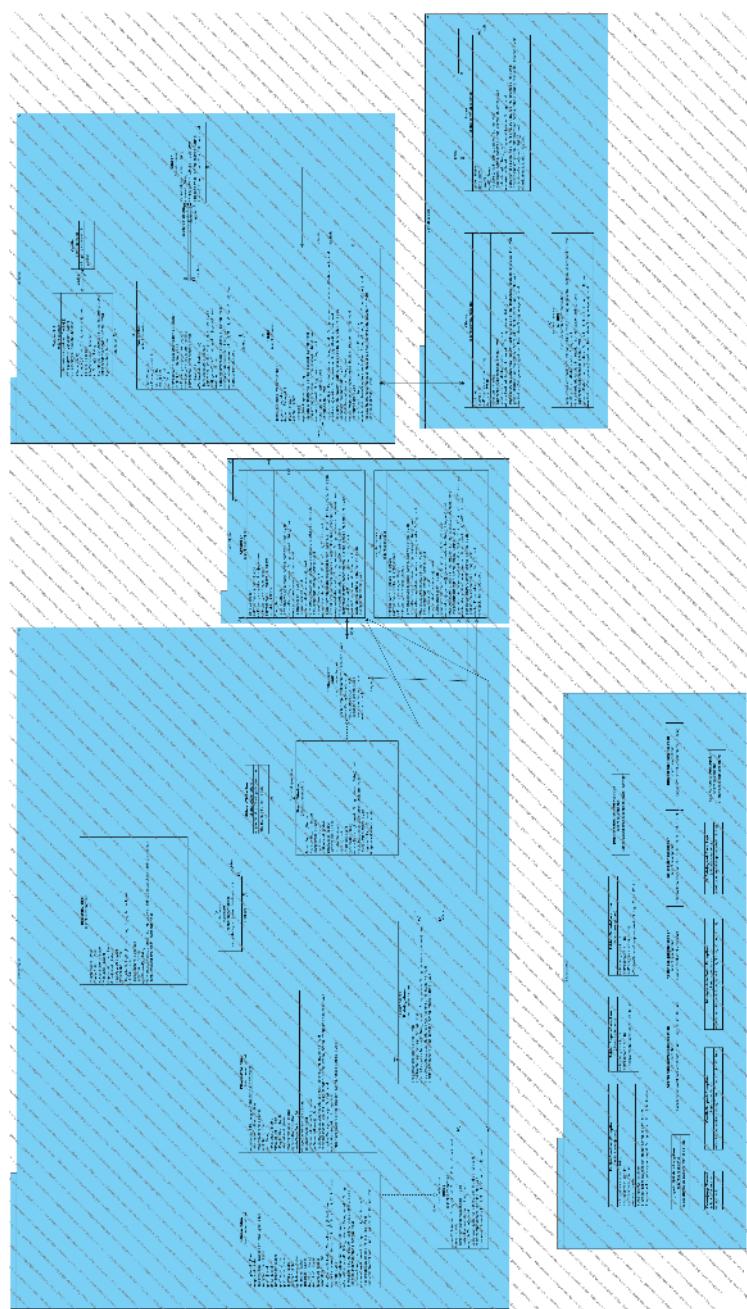


7. Diagrama de clases

7.1. Diagrama de clases del cliente

En la siguiente hoja se muestra en diagrama de clases del cliente.

7 Diagrama de clases

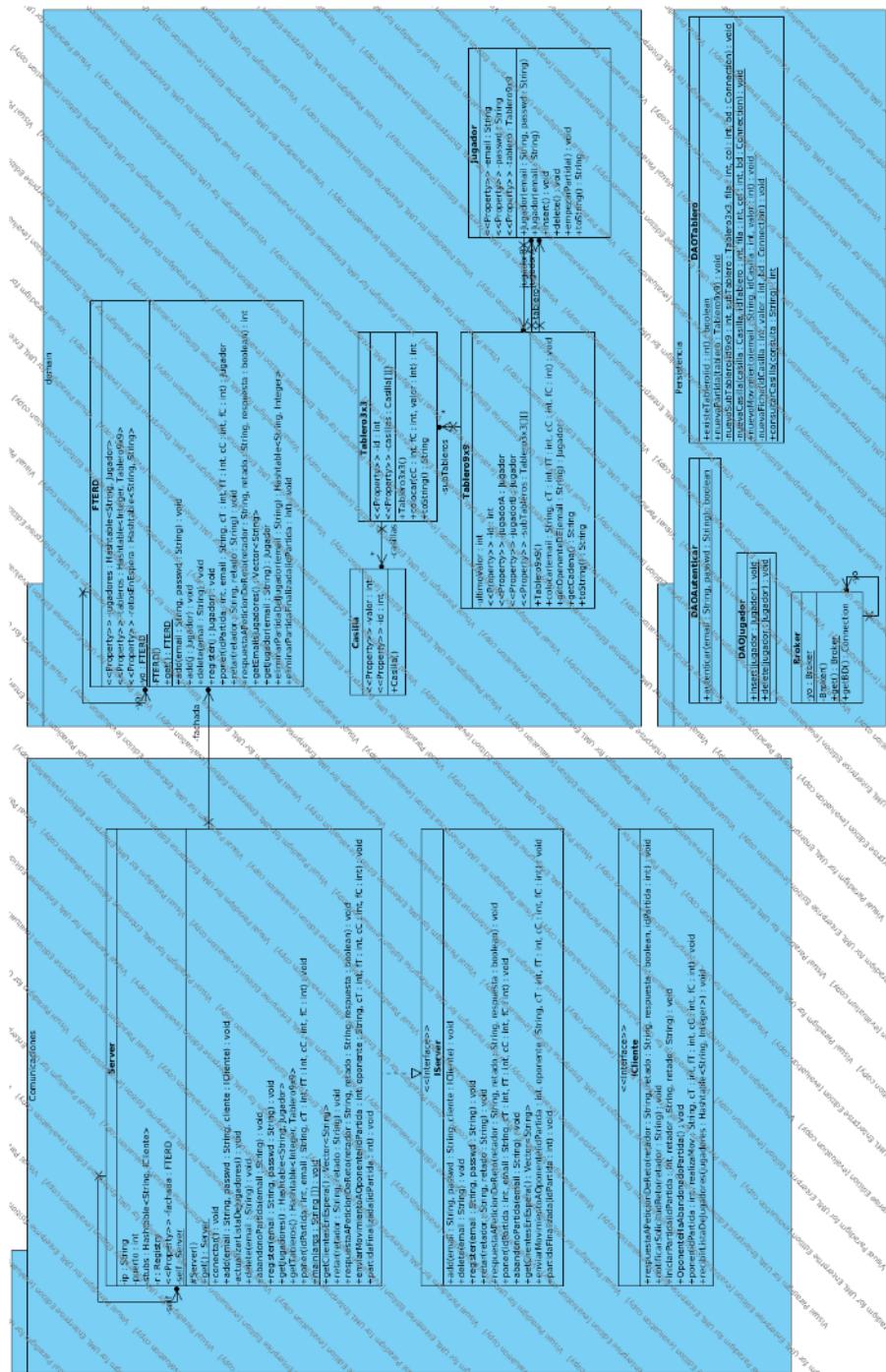


7 Diagrama de clases

7.2. Diagrama de clases del servidor

En la siguiente hoja se muestra en diagrama de clases del servidor.

7 Diagrama de clases



8. Pruebas de los casos de uso

8.1. Pruebas para el servidor

Las pruebas para el sistema del servidor se han hecho de tal forma que puedan ser ejecutadas de forma independiente. No necesita ningún cliente. El paquete `test.communications` contiene un cliente construido específicamente para las pruebas y consiste en un modelo simplificado de la clase Cliente del sistema cliente.

- Paquete `src.server.tests`

Caso de Uso	Test
Registrar Usuario	TestDAOJugador
Autenticar	TestDAOJugador
Cerrar Sesión	TestCerrarSesion
Registrar Movimiento	TestDAOMovimiento
Enviar movimiento a jugador	TestReto
Enviar respuesta a reto	TestReto

- Paquete `src.server.tests.communications`.

Caso de Uso	Test
Registrar Usuario	TestDAOJugador
Autenticar	TestDAOJugador
Cerrar Sesión	TestCerrarSesion
Registrar Movimiento	TestDAOMovimiento
Enviar movimiento a jugador	TestReto
Enviar respuesta a reto	TestReto

Los test no prueban la comunicación entre clientes, solo la funcionalidad del servidor a partir de la fachada del servidor (clase FTERD, ver Diagrama de clases del servidor).

8.2. Pruebas para el cliente