



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:** 9 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**<sup>1</sup>.

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.

---

<sup>1</sup>Este documento no explica las reglas del juego

## 2. Requisitos funcionales

### 2.1. Registrar Usuario

<b>Caso de uso</b>	Registrar Usuario
<b>Precondición</b>	Ninguna
<b>Escenario general</b>	<ol style="list-style-type: none"><li>1. Al servidor le llega una petición de registro por parte del cliente.</li><li>2. El servidor comprueba si los datos del usuario existen en la base de datos.</li><li>3. En caso de no existir los almacena en la base de datos y crea al jugador.</li><li>4. El servidor confirma al usuario que el jugador ha sido creado.</li></ol>
<b>Poscondición</b>	Un nuevo usuario existe en el sistema.

### 2.2. Iniciar sesión

<b>Caso de uso</b>	Iniciar sesión
<b>Precondición</b>	Usuario registrado
<b>Escenario general</b>	<ol style="list-style-type: none"><li>1. Al servidor le llega una petición de inicio de sesión por parte del usuario.</li><li>2. El servidor comprueba la validez de los datos del usuario.</li><li>3. El servidor confirma al gestor de usuarios que el usuario es válido.</li><li>4. El gestor de usuarios crea una sesión.</li></ol>
<b>Poscondición</b>	Se ejecuta el caso de uso “Ver lista de partidas” y usuario autenticado
<b>Escenario alternativo</b>	Puede ocurrir que los datos de acceso no sean correctos, en cuyo caso se aborta la operación y se le comunica al usuario que los datos introducidos son erróneos.

### 2.3. Cerrar sesión

<b>Caso de uso</b>	Cerrar sesión
<b>Precondición</b>	Sesión iniciada
<b>Escenario general</b>	<ol style="list-style-type: none"><li>1. Al servidor le llega una petición de cierre de sesión por parte del usuario.</li><li>2. El servidor elimina al jugador de la lista de jugadores activos.</li><li>3. Si el jugador que cierra sesión está jugando una partida, se eliminará al jugador de la partida.</li></ol>
<b>Poscondición</b>	El usuario deja de estar activo en el servidor y borrado de la lista de jugadores.

## 2 Requisitos funcionales

## 2 Requisitos funcionales

---

### 2.4. Crear Partida

<b>Caso de uso</b>	Crear Partida
<b>Precondición</b>	Ambos usuarios deben estar logeados y uno debe haber retado a otro
<b>Escenario general</b>	<ol style="list-style-type: none"><li>1. El usuario selecciona de la ventana principal crear una nueva partida.</li><li>2. La ventana principal muestra la ventana de crear partida.</li><li>3. El usuario selecciona las características de la partida.</li><li>4. La ventana crear partida habilita el botón <i>crear</i> cuando los datos son correctos.</li><li>5. El usuario pulsa el botón crear.</li><li>6. La ventana crear partida notifica al gestor de partidas que un usuario desea crear una partida.</li><li>7. El gestor de partidas pide al servidor mediante el proxy que cree una partida.</li><li>8. El proxy le indica al gestor de partidas que se ha creado una nueva partida.</li><li>9. El gestor modifica la ventana principal con el nuevo estado de las partidas.</li><li>10. La ventana principal oculta la ventana de crear partidas.</li><li>11. La ventana principal comunica al jugador que se ha creado una nueva partida.</li></ol>
<b>Poscondición</b>	En el sistema hay una nueva partida y se ejecuta el caso de uso “Unirse a partida”.
<b>Escenario alternativo</b>	<ol style="list-style-type: none"><li>1. El usuario selecciona de la ventana principal crear una nueva partida.</li><li>2. La ventana principal muestra la ventana de crear partida.</li><li>3. El usuario selecciona las características de la partida.</li><li>4. La ventana crear partida habilita el botón <i>crear</i> cuando los datos son correctos.</li><li>5. El usuario pulsa el botón crear.</li><li>6. La ventana crear partida notifica al gestor de partidas que un usuario desea crear una partida.</li><li>7. El gestor de partidas pide al servidor mediante el proxy que cree una partida.</li><li>8. El proxy le indica al gestor de partidas que no se ha podido crear la partida.</li><li>9. La ventana principal oculta la ventana de crear partidas.</li><li>10. La ventana principal comunica al jugador que no se ha podido crear la partida.</li></ol>

## 2 Requisitos funcionales

---

### 2.5. Eliminar partida

<b>Caso de uso</b>	Eliminar partida
<b>Precondición</b>	El jugador debe haber iniciado sesión y haberse unido a la partida.
<b>Escenario general</b>	<ol style="list-style-type: none"><li>1. Al servidor le llega una petición eliminar partida por parte del usuario.</li><li>2. El servidor elimina la partida.</li><li>3. EL servidor notifica a ambos jugadores la eliminación de la partida.</li></ol>
<b>Poscondición</b>	

### 2.6. Registrar Victorias

<b>Caso de uso</b>	Registrar Victorias
<b>Precondición</b>	Los usuarios deben estar conectados a una partida y la partida debe haber finalizado.
<b>Escenario general</b>	<ol style="list-style-type: none"><li>1. El servidor detecta que una partida ha finalizado y cuál es el jugador ganador.</li><li>2. El servidor almacena en la base de datos al ganador de la partida.</li><li>3. Notifica al los usuarios que la partida ha sido finalizada y qué jugador ha sido el vencedor.</li></ol>
<b>Poscondición</b>	Se elimina la partida.

### 2.7. Ver la lista de partidas

<b>Caso de uso</b>	Ver la lista de partidas
<b>Precondición</b>	Iniciar sesión
<b>Escenario general</b>	<ol style="list-style-type: none"><li>1. La ventana principal pide al gestor de partidas una lista con las partidas libres disponibles y una lista con las partidas que el usuario esta jugando.</li><li>2. El gestor de partidas pide al servidor ambas listas.</li><li>3. El servidor envía ambas listas de partidas al gestor de partidas.</li><li>4. La ventana principal muestra la lista de partidas disponibles y la lista de partidas comenzadas.</li></ol>
<b>Poscondición</b>	La lista de partidas esta disponible para el usuario.

## 2 Requisitos funcionales

---

### 2.8. Enviar Petición de Reto

<b>Caso de uso</b>	Enviar petición de Reto
<b>Precondición</b>	Usuarios logeados en el sistema y petición de reto realizada.
<b>Escenario general</b>	<ol style="list-style-type: none"><li>1. Al servidor le llega una petición de reto por parte de un usuario.</li><li>2. El servidor envía al otro jugador la petición de reto.</li></ol>
<b>Poscondición</b>	

### 2.9. Registrar un movimiento

<b>Caso de uso</b>	Registrar un movimiento
<b>Precondición</b>	Iniciar sesión, Unirse a una partida y Conectarse a una partida
<b>Escenario general</b>	<ol style="list-style-type: none"><li>1. Al servidor le llega el movimiento realizado por un usuario.</li><li>2. El servidor almacena el movimiento en la base de datos.</li><li>3. El servidor notifica al otro usuario que ya puede realizar su movimiento.</li></ol>
<b>Poscondición</b>	Se realiza el caso de uso: "Enviar actualización de la partida".

### 2.10. Responder a solicitud de reto

<b>Caso de uso</b>	Responder a solicitud de reto
<b>Precondición</b>	Iniciar sesión
<b>Escenario general</b>	<ol style="list-style-type: none"><li>1. El servidor envía al usuario la solicitud de reto del otro usuario.</li><li>2. El cliente recibe la solicitud y la envía al controlador.</li><li>3. El controlador la envía a la ventana de jugadores para que muestre que el jugador ha sido retado</li><li>4. El usuario elige la acción en la ventana (si acepta o si no).</li><li>5. La ventana de juego envía la información al controlador.</li><li>6. La fachada envía la respuesta al servidor por medio del proxy.</li></ol>
<b>Poscondición</b>	Respuesta a reto enviada.

### 2.11. Realizar movimiento

<b>Caso de uso</b>	Realizar Movimiento
<b>Precondición</b>	El usuario debe haberse unido a una partida y estar conectado a ella.
<b>Escenario general</b>	<ol style="list-style-type: none"><li>1. El usuario pulsa sobre la casilla del tablero a colocar.</li><li>2. La interfaz avisa al controlador del movimiento.</li><li>3. El controlador envía el movimiento a la fachada.</li><li>4. La fachada envía el movimiento a Tablero9x9.</li><li>5. Tablero9x9 comprueba la legalidad del movimiento.</li><li>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.</li><li>7. Si el movimiento es inválido se lanza una excepción que se recoge en el controlador.</li><li>8. El controlador envía la excepción a la interfaz de usuario para comunicarle el error que ha cometido.</li></ol>
<b>Postcondición</b>	Movimiento realizado.

### 2.12. Enviar petición de alianza

<b>Caso de uso</b>	Enviar petición de alianza
<b>Precondición</b>	El usuario debe haberse unido a una partida
<b>Escenario general</b>	<ol style="list-style-type: none"> <li>1. El usuario pulsa sobre el botón alianza.</li> <li>2. Se muestra al usuario una ventana con diferentes jugadores.</li> <li>3. El usuario selecciona uno de ellos y confirma la operación.</li> <li>4. El gestor del juego recibe los datos y envía la información al proxy.</li> <li>5. El proxy notifica al gestor de juego que la alianza se ha realizado con éxito.</li> <li>6. El gestor de juego actualiza los datos del juego.</li> </ol>
<b>Postcondición</b>	Mostrar correctamente los datos de juego una vez actualizado y se ejecuta el caso de uso “Enviar actualización de la partida”.
<b>Escenario alternativo</b>	<ol style="list-style-type: none"> <li>1. El usuario pulsa sobre el botón alianza.</li> <li>2. Se muestra al usuario una ventana con diferentes jugadores.</li> <li>3. El usuario selecciona uno de ellos y confirma la operación.</li> <li>4. El gestor del juego recibe los datos, comprueba que se cumple la condición que permite la alianza, pero el jugador seleccionado no es válido para una alianza.</li> <li>5. El gestor de juego pide a la ventana que muestre un mensaje con información sobre el error producido al realizar la alianza.</li> </ol>

## 2 Requisitos funcionales

---

### 2.13. Responder a petición de alianza

<b>Caso de uso</b>	Responder a petición de alianza
<b>Precondición</b>	El usuario debe haberse unido a una partida
<b>Escenario general</b>	<ol style="list-style-type: none"><li>1. El proxy comunica al gestor de juego que existe una petición de alianza.</li><li>2. El gestor de juego pide a la ventana que muestre al jugador el mensaje pertinente.</li><li>3. El usuario selecciona la acción oportuna.</li><li>4. El gestor del juego recibe los datos y envía la información al proxy.</li><li>5. El proxy notifica al gestor de juego que la alianza se ha realizado con éxito.</li><li>6. El gestor de juego actualiza los datos del juego.</li></ol>
<b>Postcondición</b>	Mostrar correctamente los datos de juego una vez actualizado Se ejecuta el caso de uso “Enviar actualización de la partida”. .

### 2.14. Romper alianza

<b>Caso de uso</b>	Romper alianza
<b>Precondición</b>	El usuario debe haberse unido a una partida y tener, al menos, una alianza
<b>Escenario general</b>	<ol style="list-style-type: none"><li>1. El usuario pulsa sobre el botón alianza.</li><li>2. Se muestra al usuario una ventana con diferentes alianzas.</li><li>3. El usuario selecciona una de ellas y confirma la operación.</li><li>4. El gestor del juego recibe los datos y envía la información al proxy.</li><li>5. El proxy notifica al gestor de juego que la alianza se ha roto con éxito.</li><li>6. El gestor de juego actualiza los datos del juego.</li></ol>
<b>Postcondición</b>	Mostrar correctamente los datos de juego una vez actualizado. Se ejecuta el caso de uso “Enviar actualización de la partida”. .

### 3 Otros requisitos

---

#### 2.15. Enviar actualización de la partida

<b>Caso de uso</b>	Enviar actualización de la partida
<b>Precondición</b>	El usuario debe haberse unido a una partida
<b>Escenario general</b>	<ol style="list-style-type: none"><li>1. El usuario realiza una operación que modifica la partida.</li><li>2. El gestor de juego recopila todos los datos modificados y envía la información al proxy.</li><li>3. El proxy notifica al gestor de juego que la actualización se ha realizado con éxito.</li></ol>
<b>Postcondición</b>	Los datos del cliente y del servidor están sincronizados.

#### 2.16. Recibir actualización de la partida

<b>Caso de uso</b>	Recibir actualización de la partida
<b>Precondición</b>	El usuario debe haberse unido a una partida
<b>Escenario general</b>	<ol style="list-style-type: none"><li>1. El proxy comunica al gestor de juego que la partida ha cambiado.</li><li>2. El gestor de juego recibe todos los datos modificados y pide a la ventana que se actualice.</li></ol>
<b>Postcondición</b>	Los datos del cliente y del servidor están sincronizados.

### 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:

### 3 Otros requisitos

---

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:

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

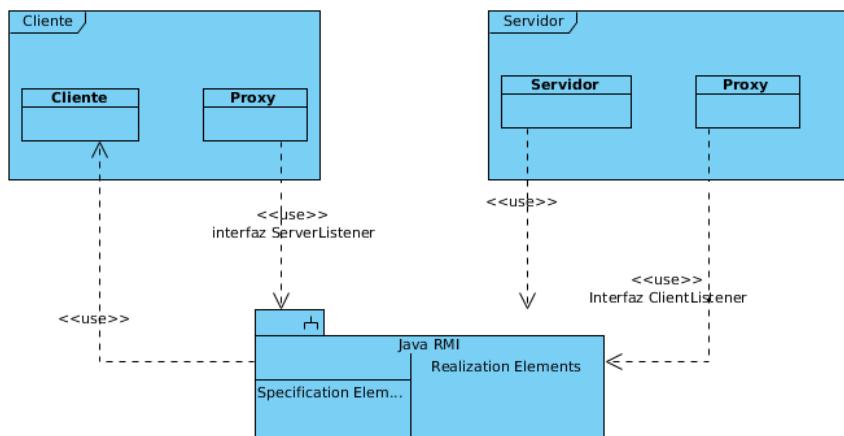
**Descripción:** Muestra al usuario todos los detalles de una partida y permite jugar en ella. Deberá mostrar el tablero de juego con la asignación de territorios.

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.

## 4. Arquitectura del sistema

A continuación 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.

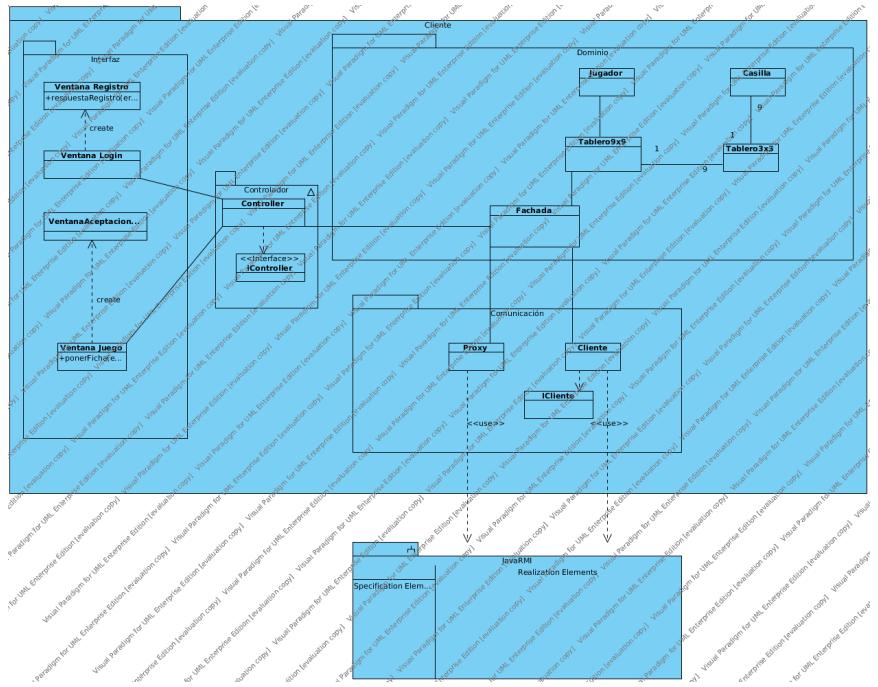


### 4.1. Arquitectura del cliente

A continuación se muestra la arquitectura del cliente. Esta es una arquitectura mult capa en el que existen varias capas:

- *Interfaz*: Este paquete contiene las clases de las ventanas que se muestran al usuario, y las interfaces que estas implementan.
- *Controlador*: Este paquete contiene el controlador y su interfaz. Realiza las comunicaciones y el control de la interfaz gráfica con el dominio.
- *Dominio*: Contiene toda la lógica del juego. La clase Tablero9x9 contiene todas las reglas del juego y el modelo de la partida actual. La clase Fachada realiza las funciones de paso de mensajes entre las capas de control y comunicación, además de realizar las acciones consideradas en el modelo del juego.
- *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 clases solamente se encargan del envío de los respectivos mensajes al servidor o a la fachada del paquete de dominio.

## 4 Arquitectura del sistema



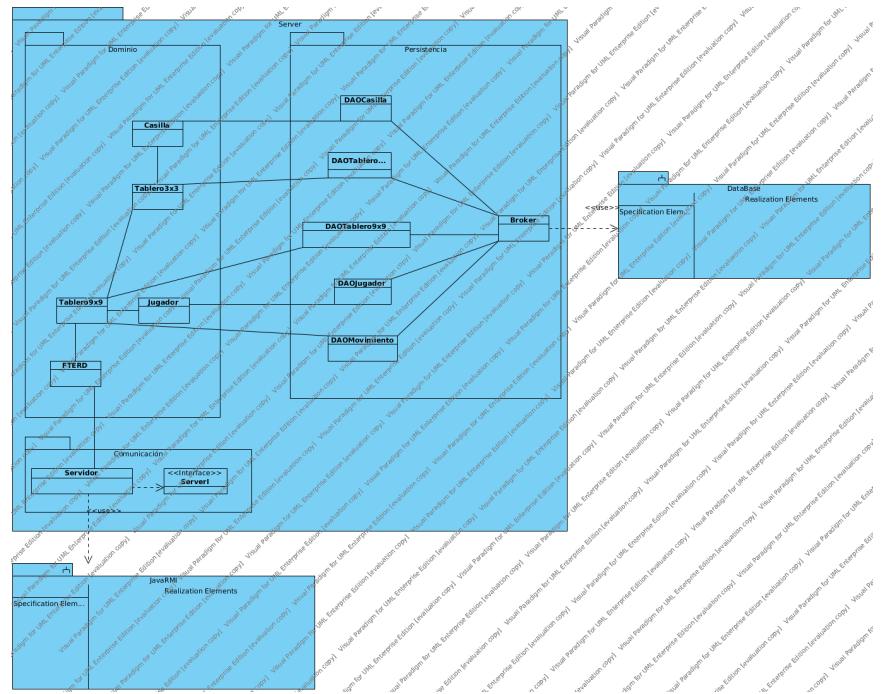
### 4.2. Arquitectura del servidor

A continuación se muestra la arquitectura del servidor. Esta es una arquitectura multicapa en la que constan los siguientes paquetes:

- **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. Además se pueden observar las clases DAO, encargadas del almacenamiento de cada una de las entidades pertenecientes al dominio del juego.
- **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.

## 4 Arquitectura del sistema

---

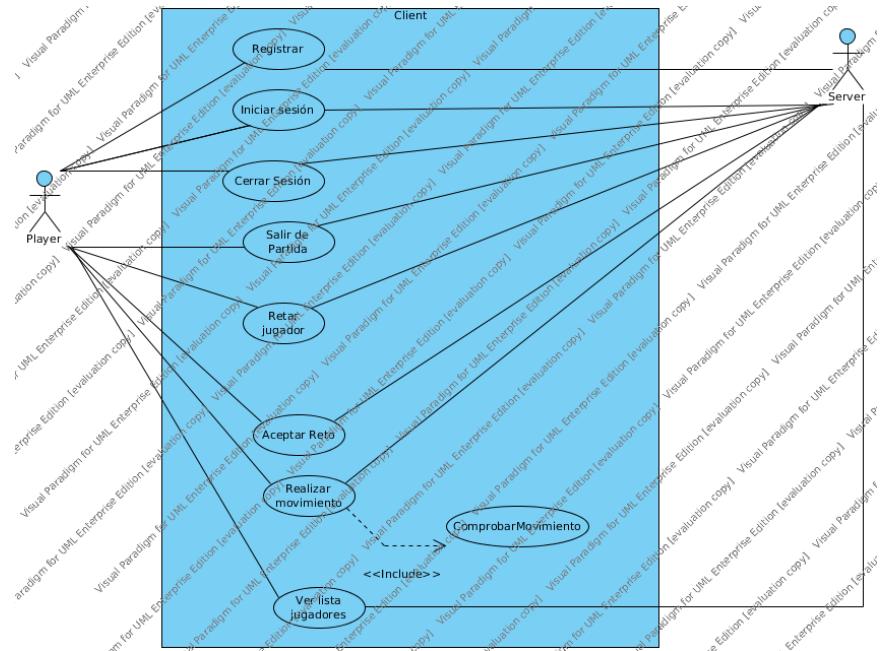


## 5 Casos de uso

---

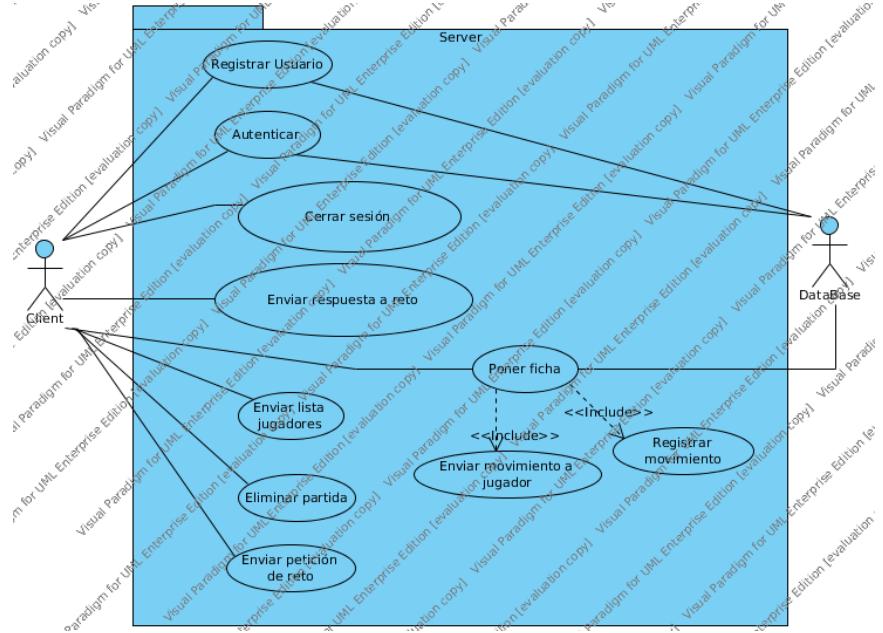
### 5. Casos de uso

#### 5.1. Diagrama de casos de uso del cliente



## 6 Diagramas de secuencia

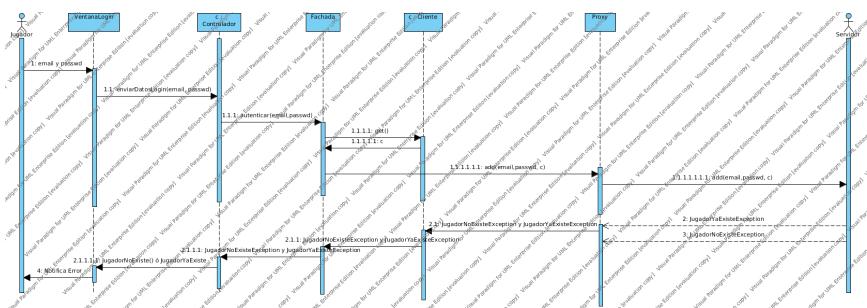
### 5.2. Diagrama de casos de uso del servidor



## 6. Diagramas de secuencia

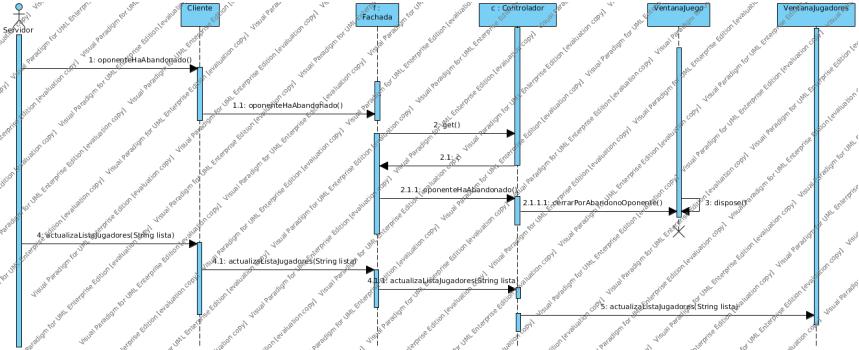
### 6.1. Diagramas de secuencia del cliente

#### 6.1.1. Autenticar

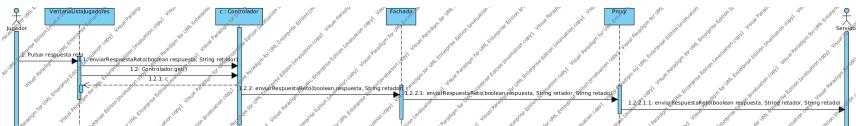


## 6 Diagramas de secuencia

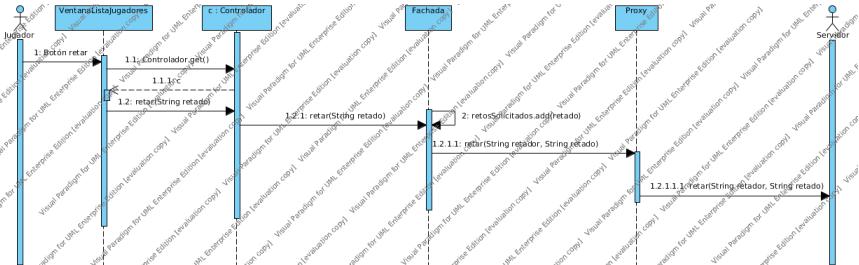
### 6.1.2. Cerrar sesión oponente



### 6.1.3. Enviar respuesta de reto



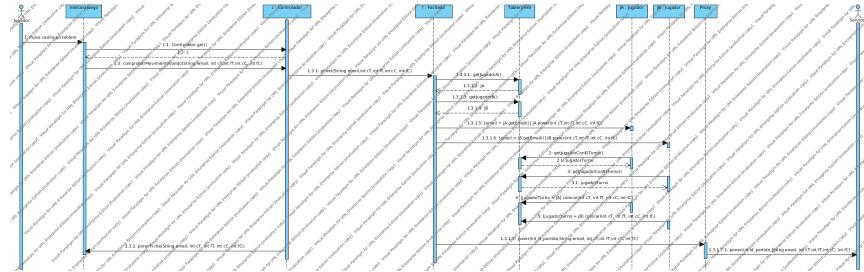
### 6.1.4. Enviar reto



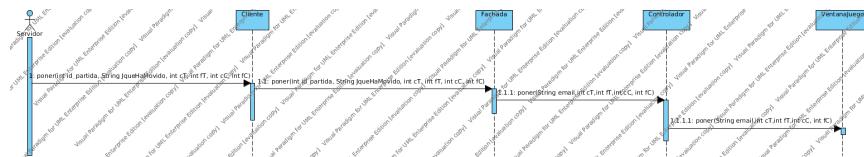
## 6 Diagramas de secuencia

---

### 6.1.5. Realizar movimiento

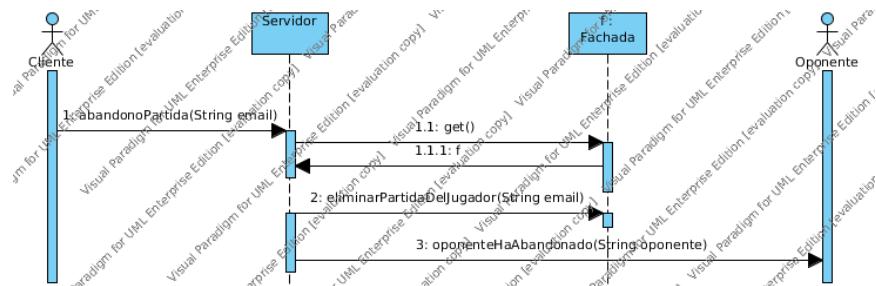


### 6.1.6. Recibir movimiento



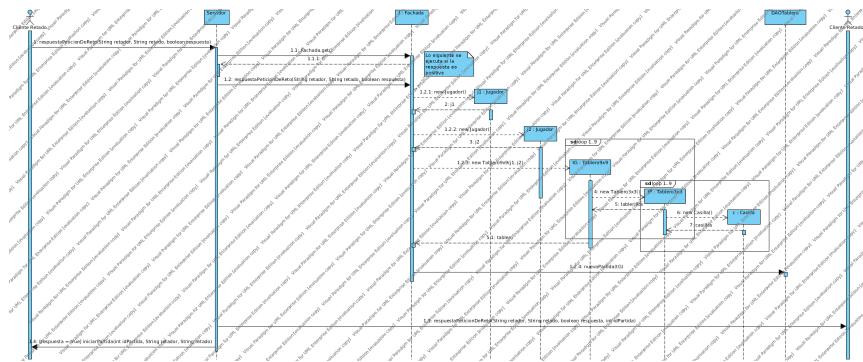
## 6.2. Diagramas de secuencia del servidor

### 6.2.1. Cerrar sesión

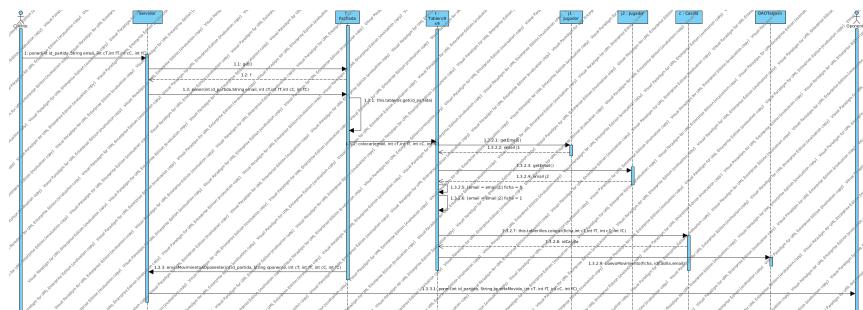


## 6 Diagramas de secuencia

### **6.2.2. Enviar respuesta de reto**



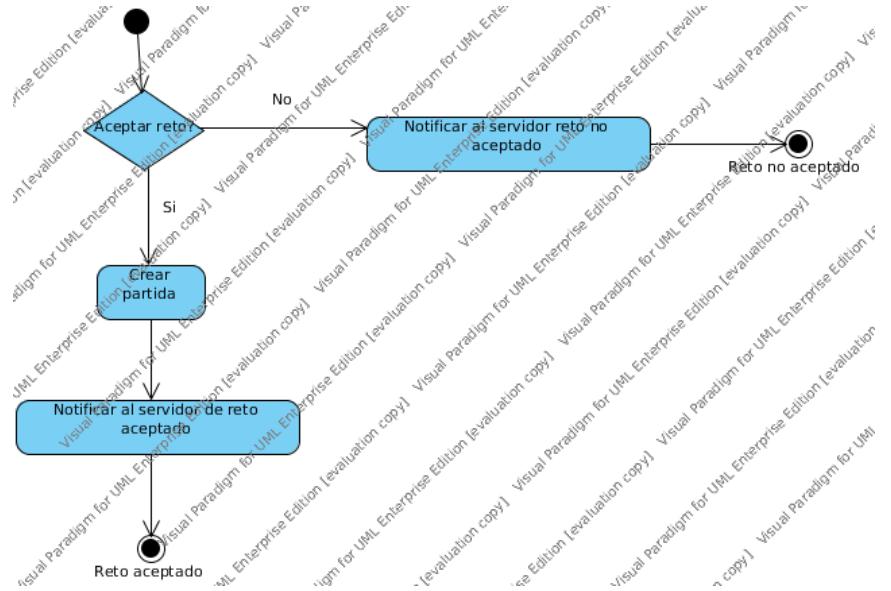
### **6.2.3. Realizar movimiento**



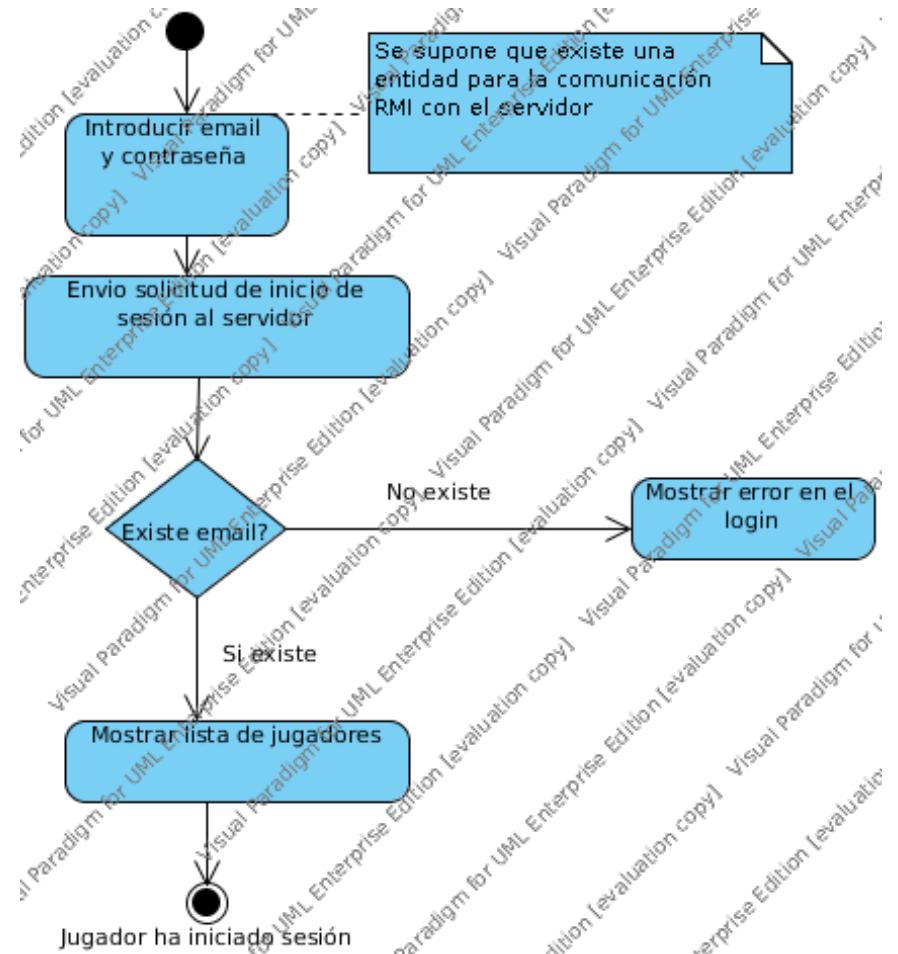
## 7. Máquinas de estado

### 7.1. Máquinas de estado del cliente

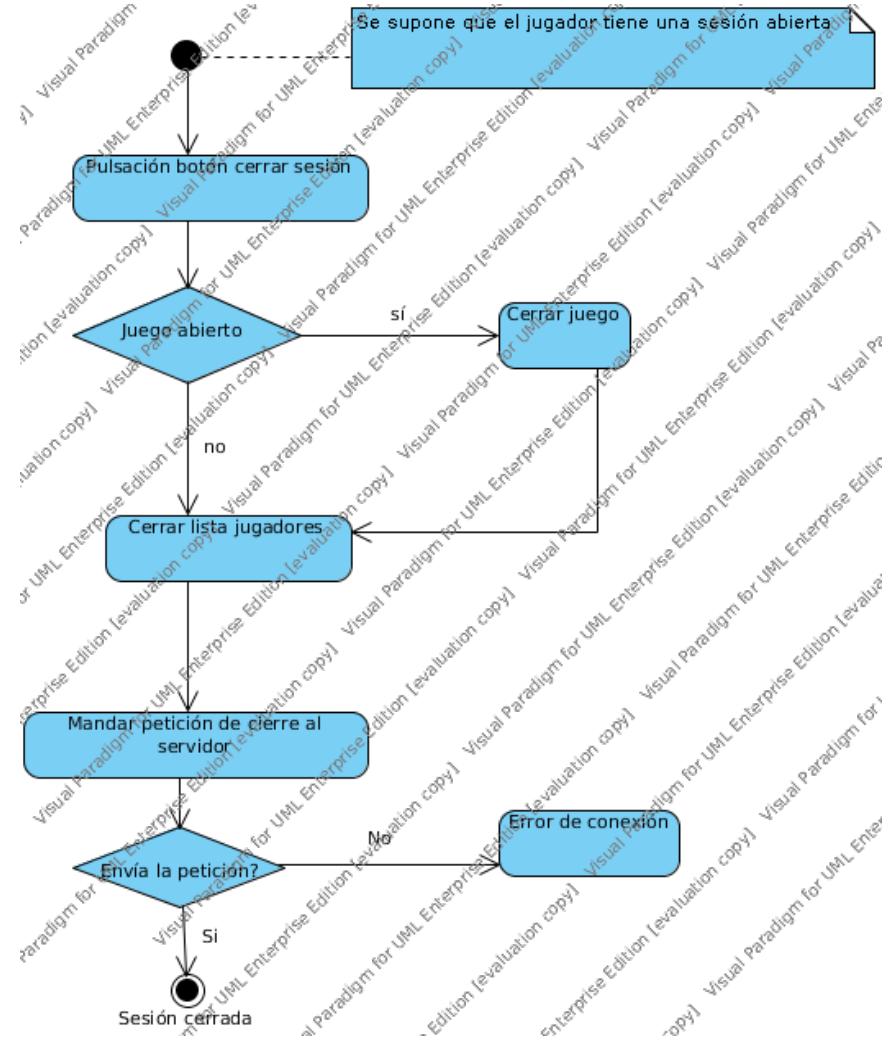
#### 7.1.1. Aceptar reto



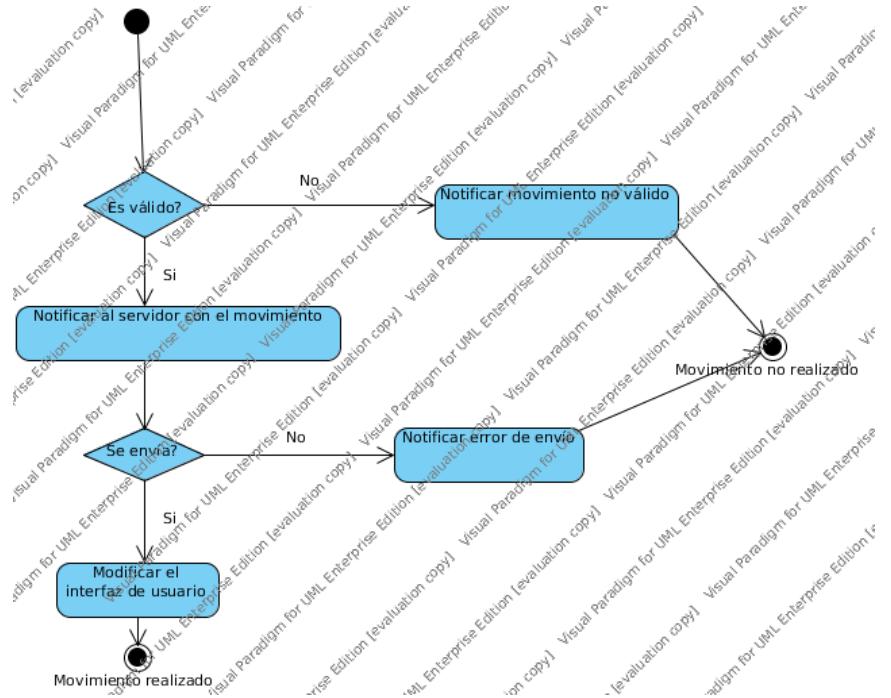
### 7.1.2. Iniciar sesión



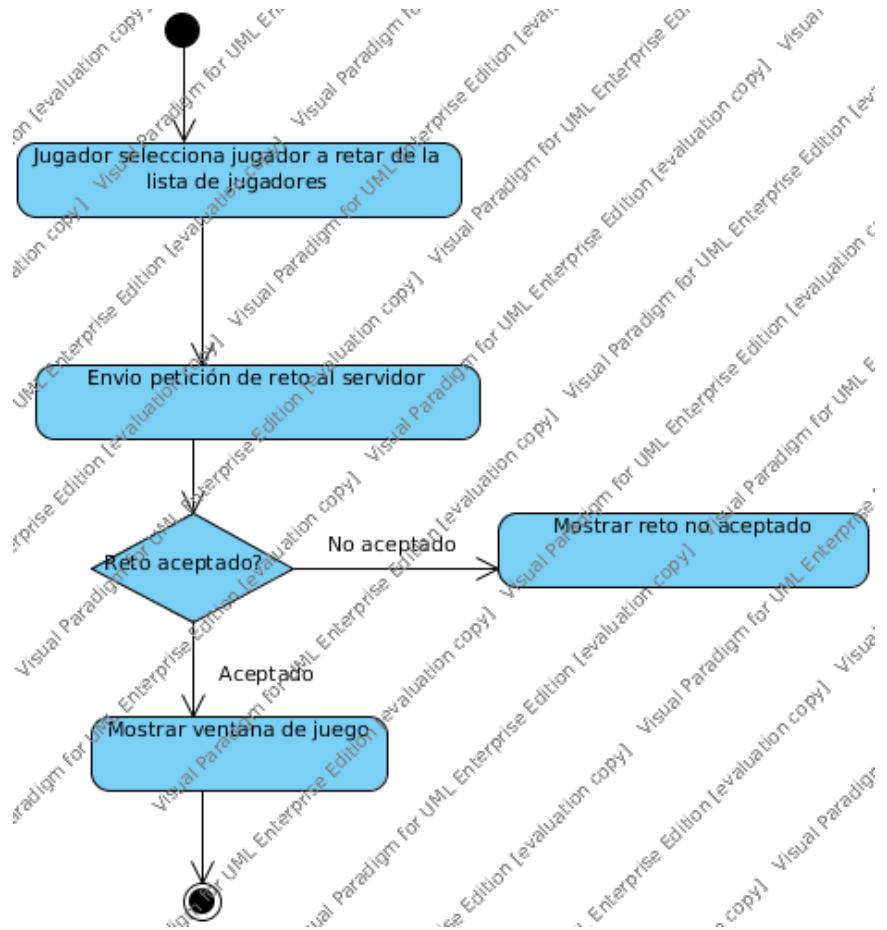
### 7.1.3. Cerrar sesión



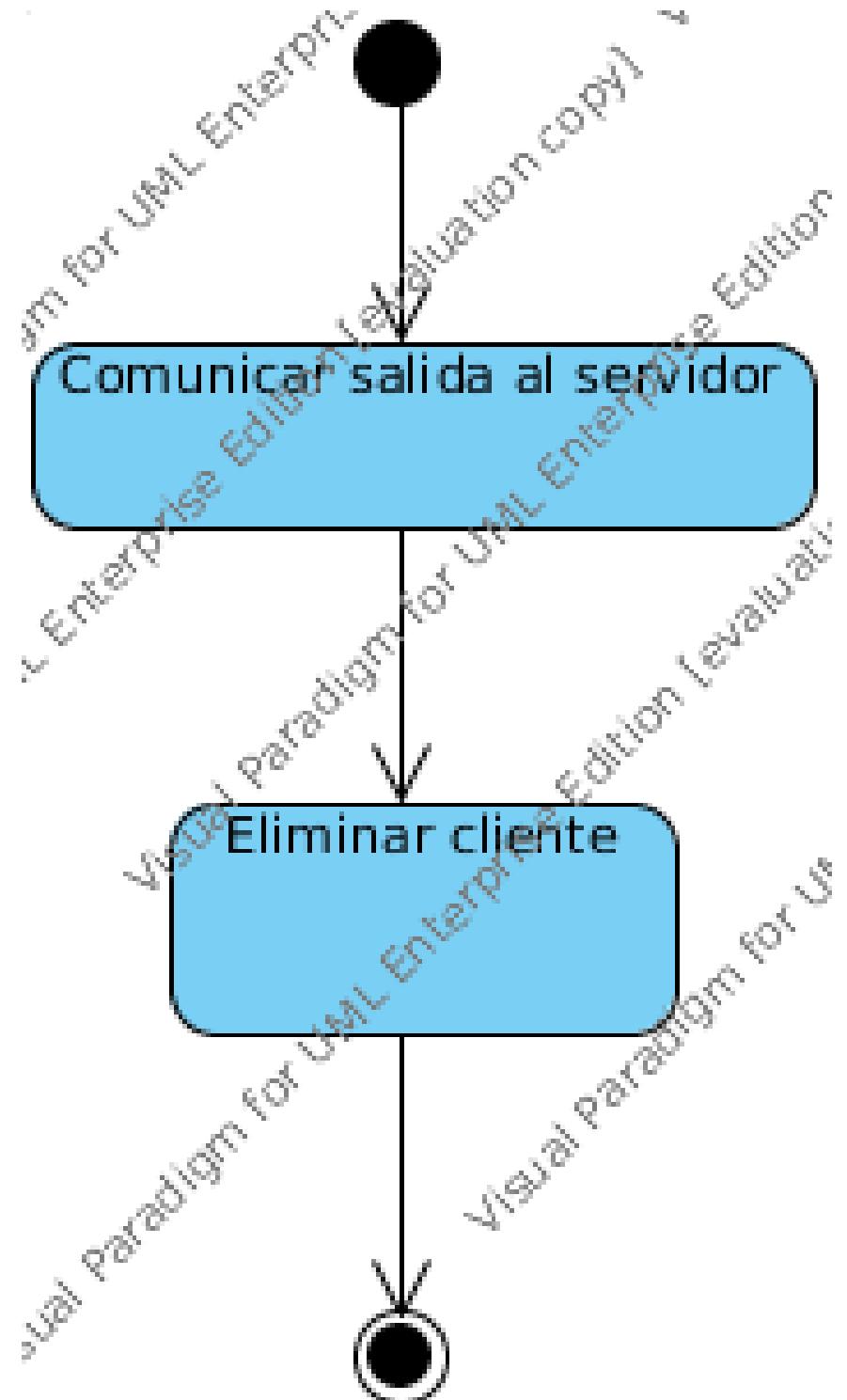
#### 7.1.4. Realizar movimiento



### 7.1.5. Retar jugador

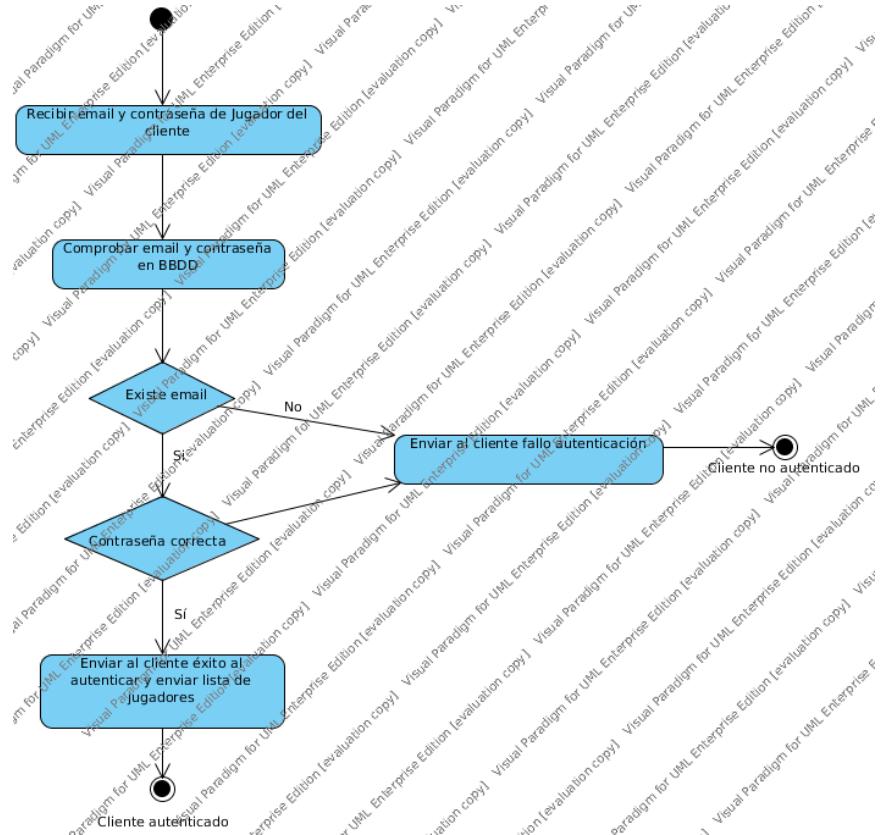


#### 7.1.6. Salir de partida

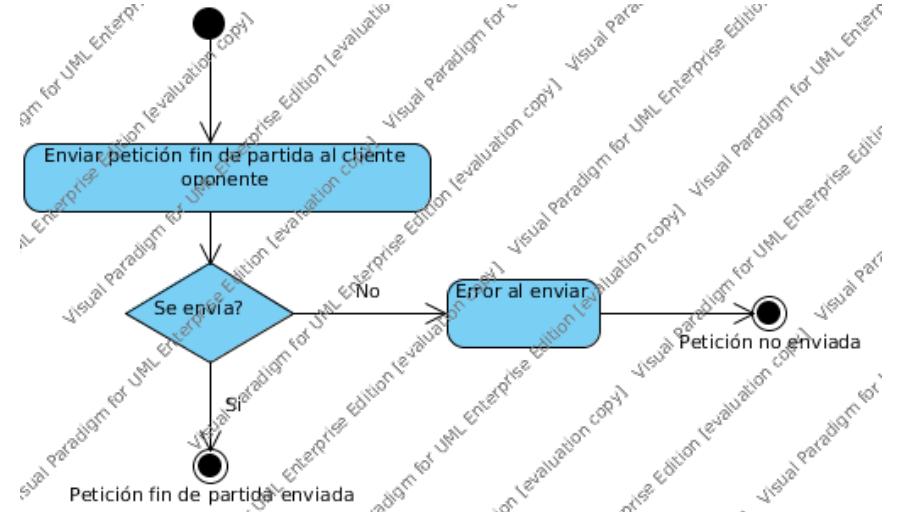


## 7.2. Máquinas de estado del Servidor

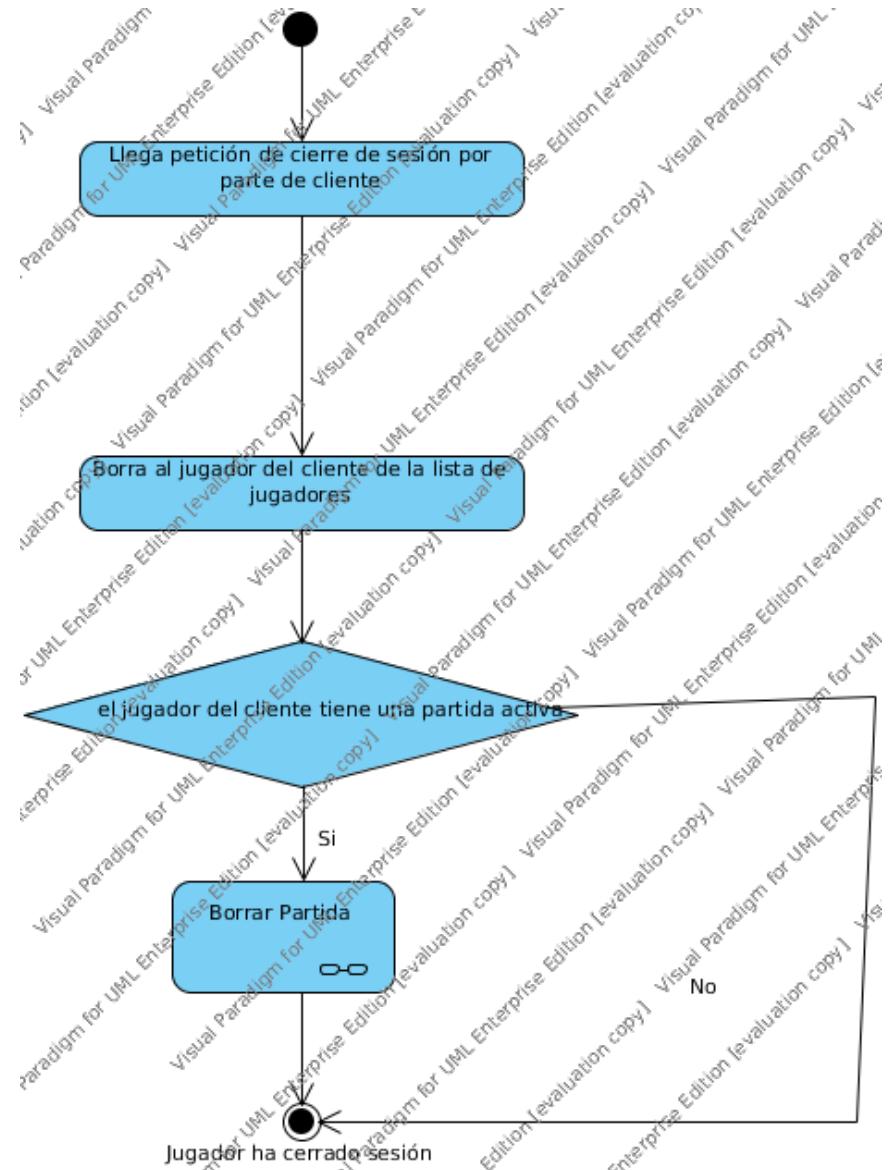
### 7.2.1. Autenticar



### 7.2.2. Borrar partida



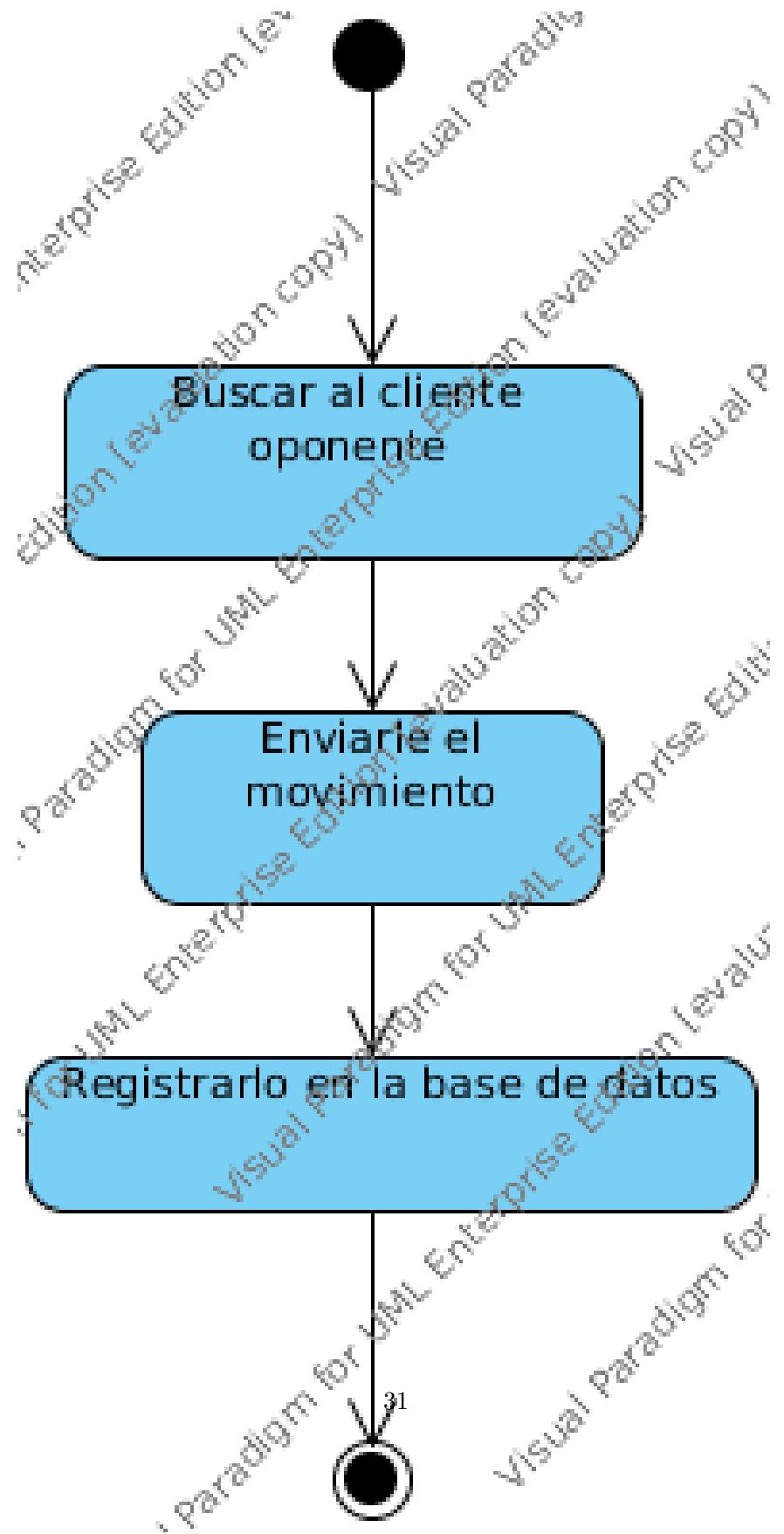
### 7.2.3. Cerrar sesión



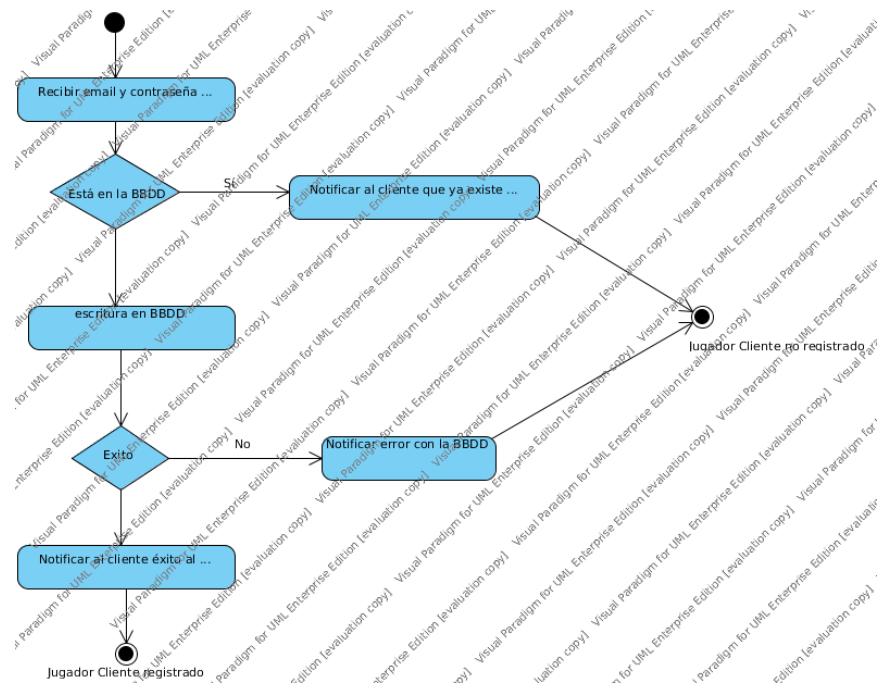
## 7 Máquinas de estado

---

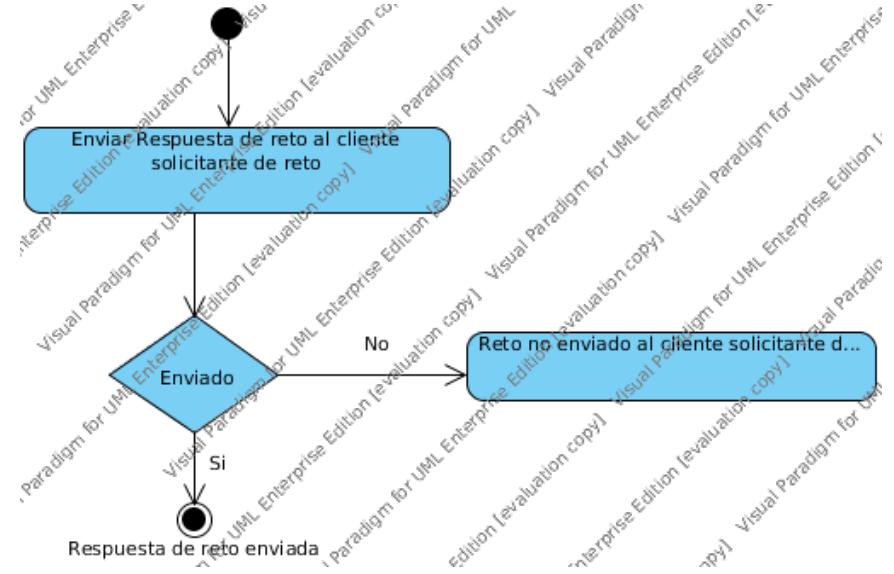
7.2.4. Poner ficha



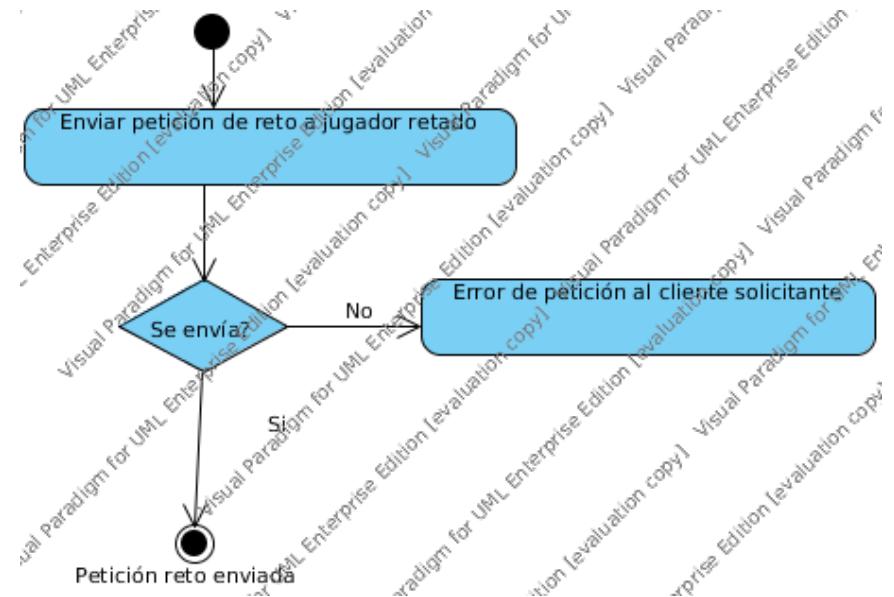
### 7.2.5. Registrar usuario



### 7.2.6. Respuesta de reto



### 7.2.7. Retar jugador

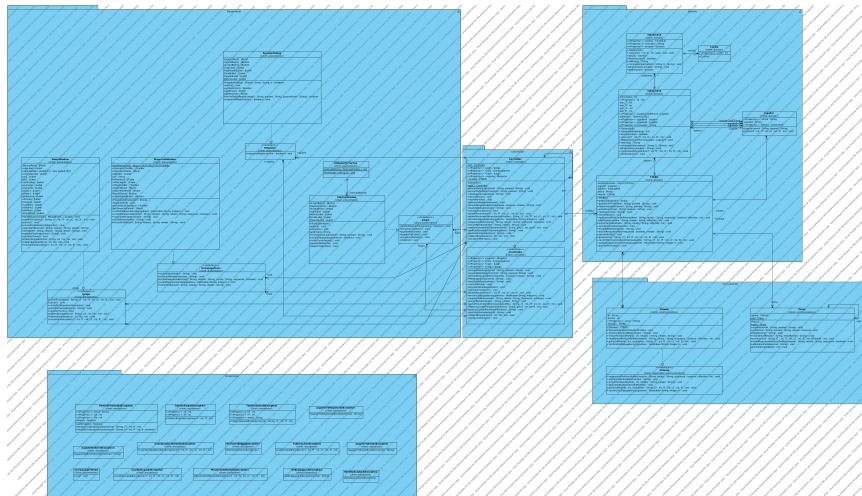


## 8 Diagrama de clases

## 8. Diagrama de clases

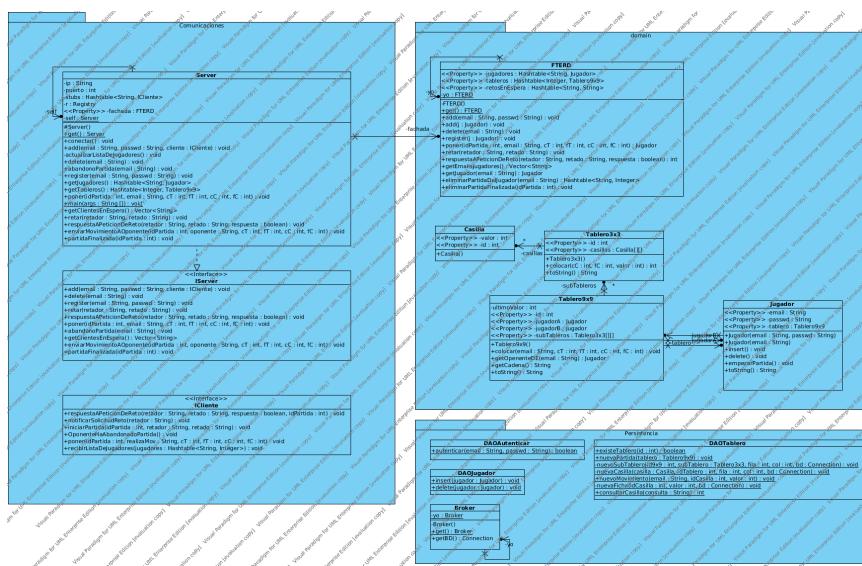
En esta sección se realizará un análisis más en profundidad del diseño tanto del cliente como del servidor en cuanto a sus clases se refiere.

### 8.1. Diagrama de clases del cliente



Como

## 8.2. Diagrama de clases del servidor



## **9. Pruebas de los casos de uso**