



ESCUELA SUPERIOR DE INFORMÁTICA  
UNIVERSIDAD DE CASTILLA – LA MANCHA

Práctica 1

**Tres en raya definitivo**

*Autores*

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

**Asignatura:** Ingeniería del Software II  
**Titulación:** Ingeniería Informática  
**Fecha:** 10 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

### 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

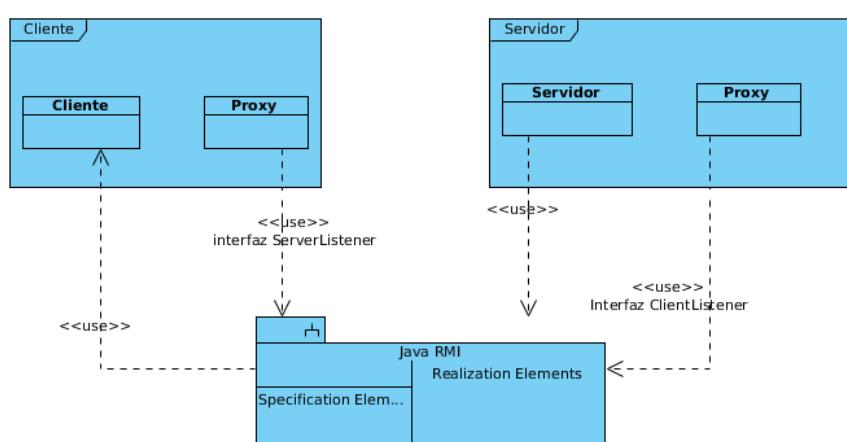


Figura 1: Arquitectura del sistema

#### 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

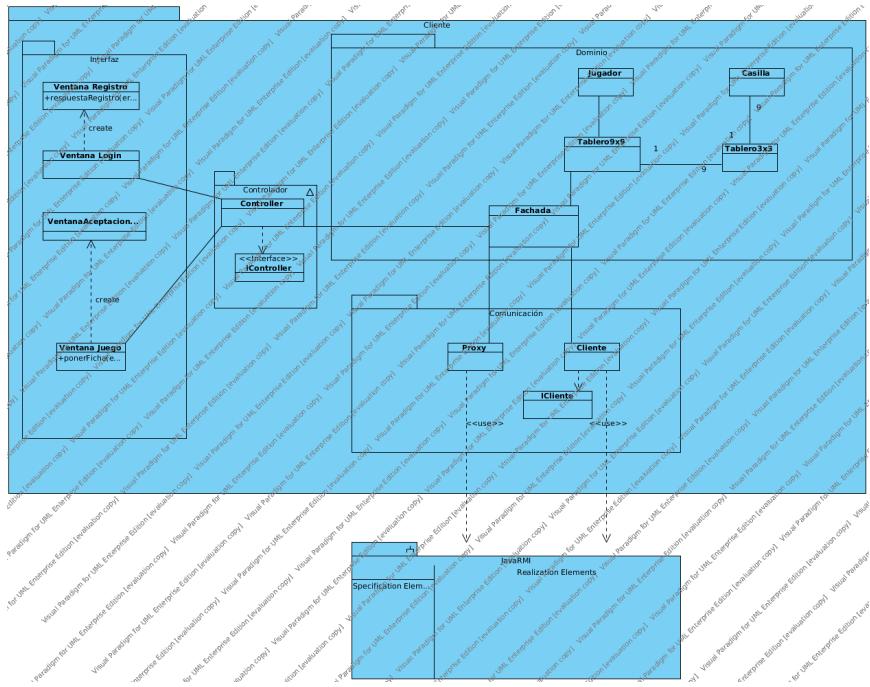


Figura 2: Arquitectura del cliente

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.

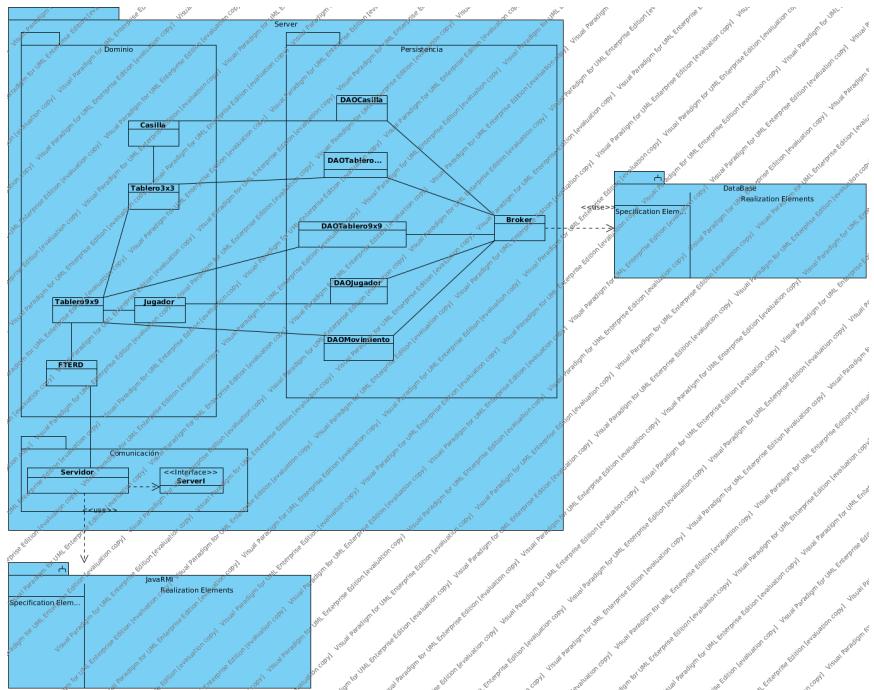


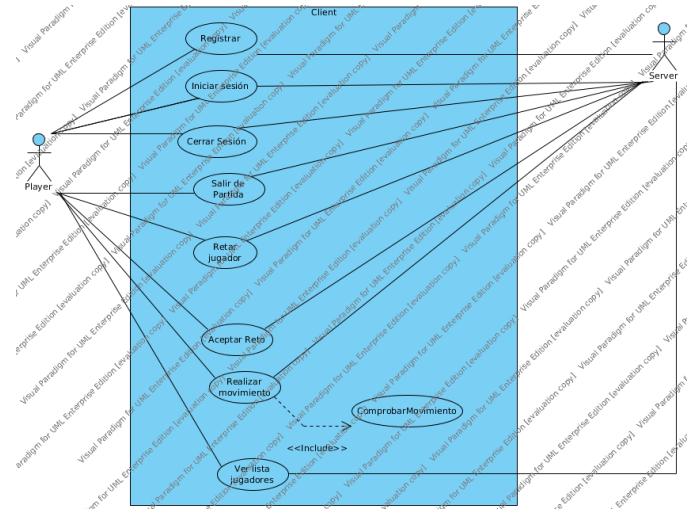
Figura 3: Arquitectura del servidor

## 4 Casos de uso

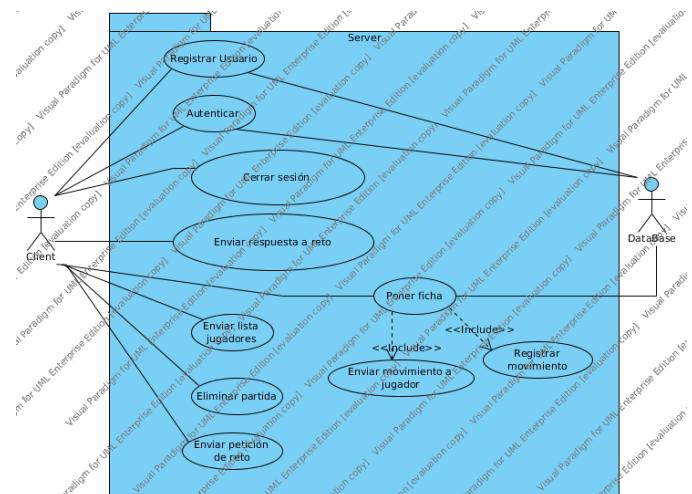
---

### 4. Casos de uso

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



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



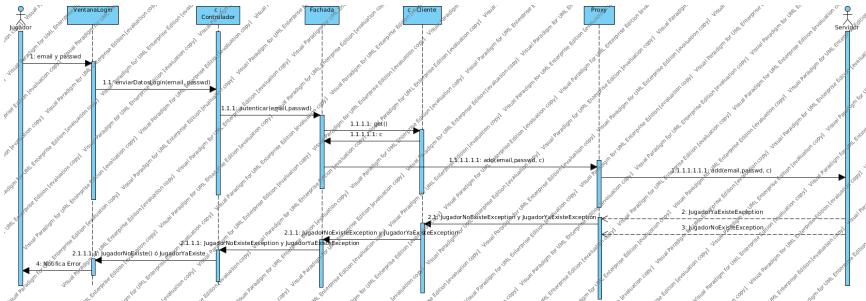
## 5 Diagramas de secuencia

---

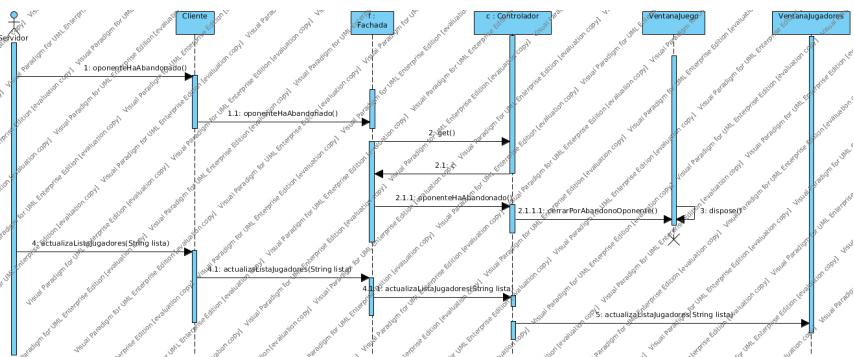
### 5. Diagramas de secuencia

#### 5.1. Diagramas de secuencia del cliente

##### 5.1.1. Autenticar



##### 5.1.2. Cerrar sesión oponente

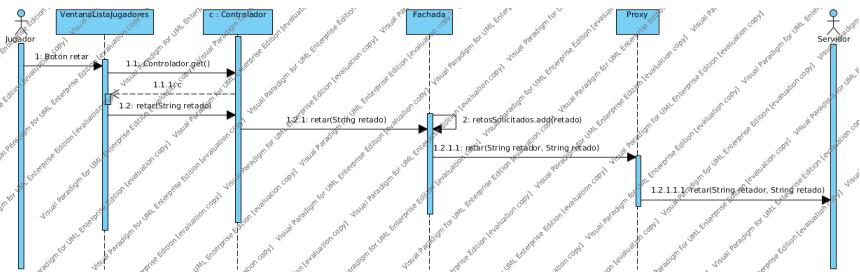


##### 5.1.3. Enviar respuesta de reto

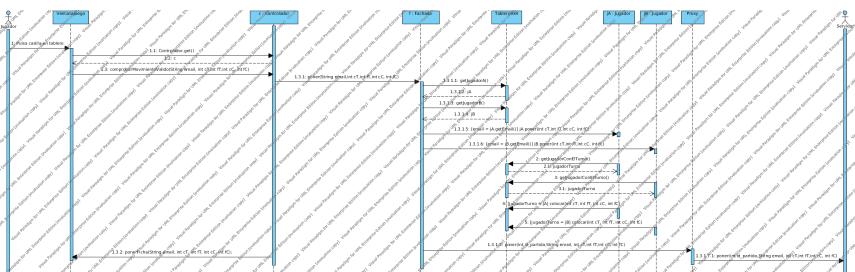


## 5 Diagramas de secuencia

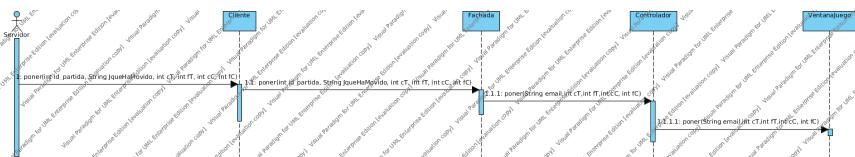
#### 5.1.4. Enviar reto



### 5.1.5. Realizar movimiento



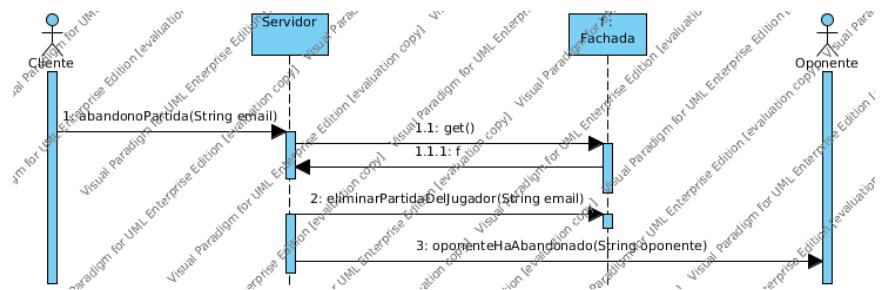
### 5.1.6. Recibir movimiento



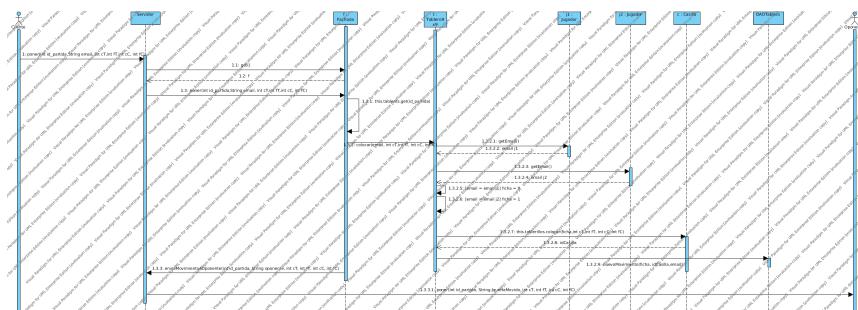
## 5 Diagramas de secuencia

## 5.2. Diagramas de secuencia del servidor

### 5.2.1. Cerrar sesión

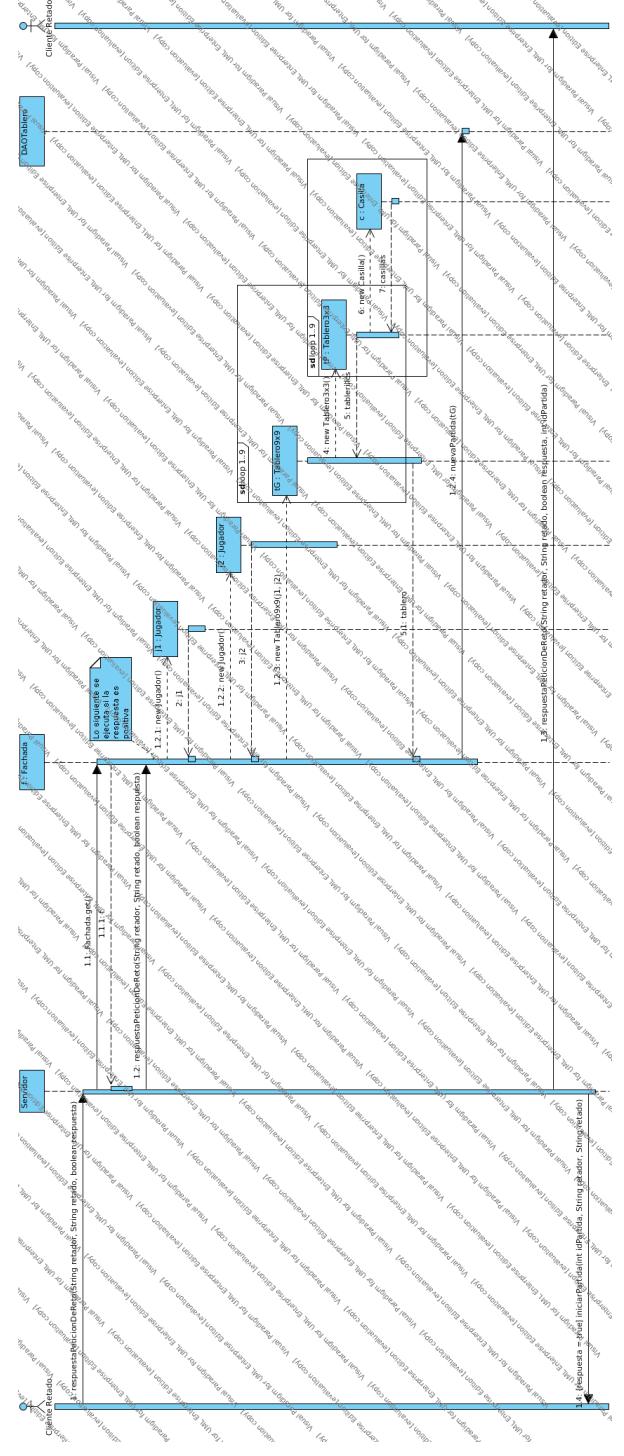


### **5.2.2. Realizar movimiento**



## 5 Diagramas de secuencia

### 5.2.3. Enviar respuesta de reto

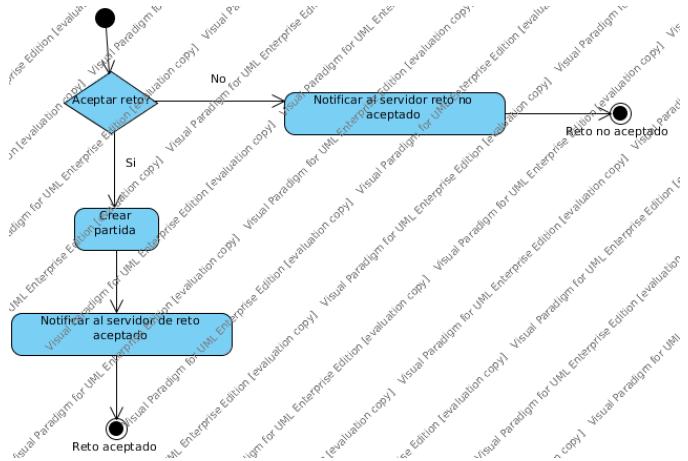


## 6 Máquinas de estado

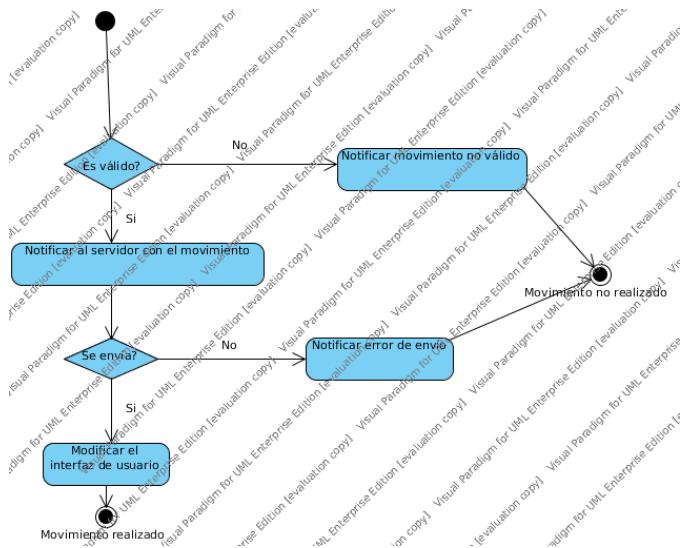
### 6. Máquinas de estado

#### 6.1. Máquinas de estado del cliente

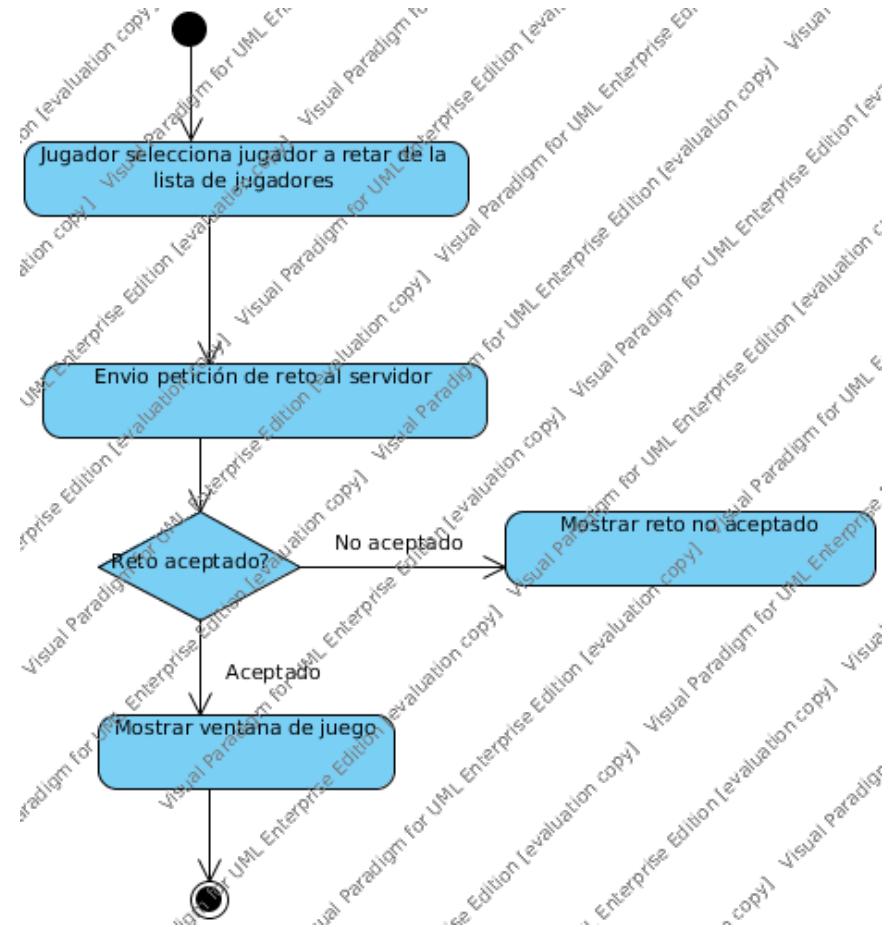
##### 6.1.1. Aceptar reto



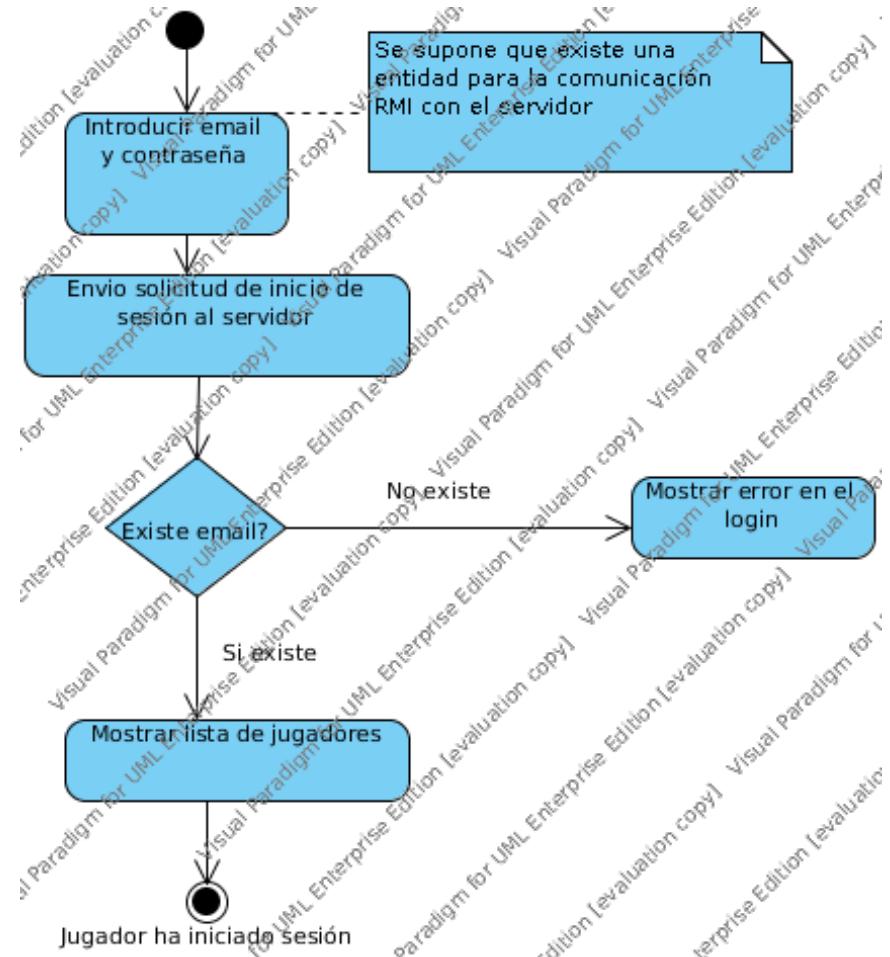
##### 6.1.2. Realizar movimiento



### 6.1.3. Retar jugador

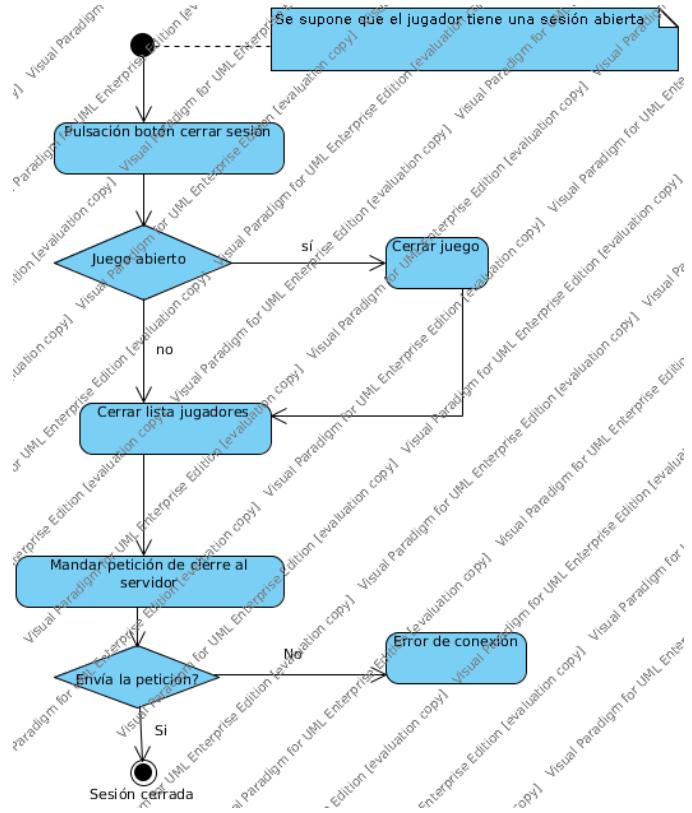


#### 6.1.4. Iniciar sesión

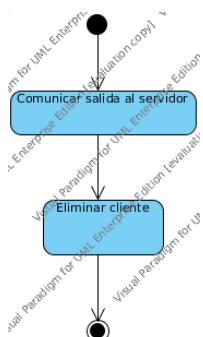


## 6 Máquinas de estado

### 6.1.5. Cerrar sesión



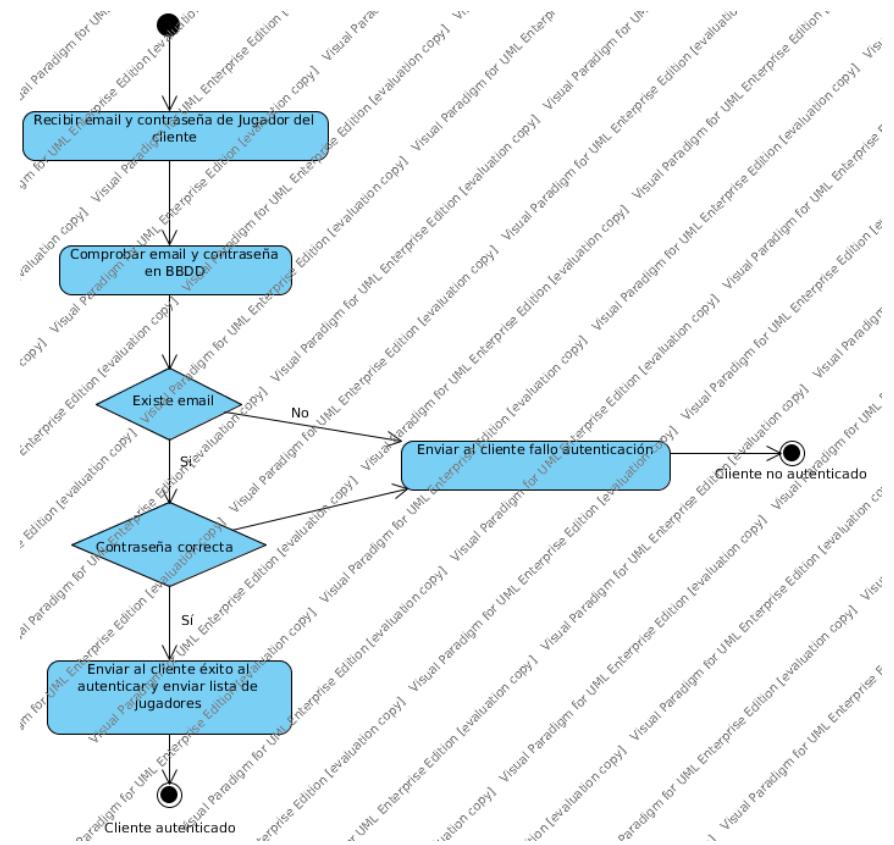
### 6.1.6. Salir de partida



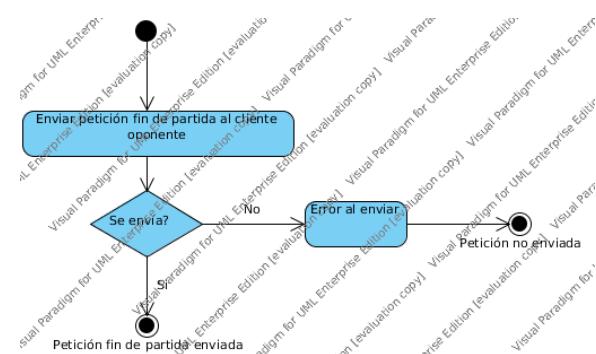
## 6 Máquinas de estado

### 6.2. Máquinas de estado del Servidor

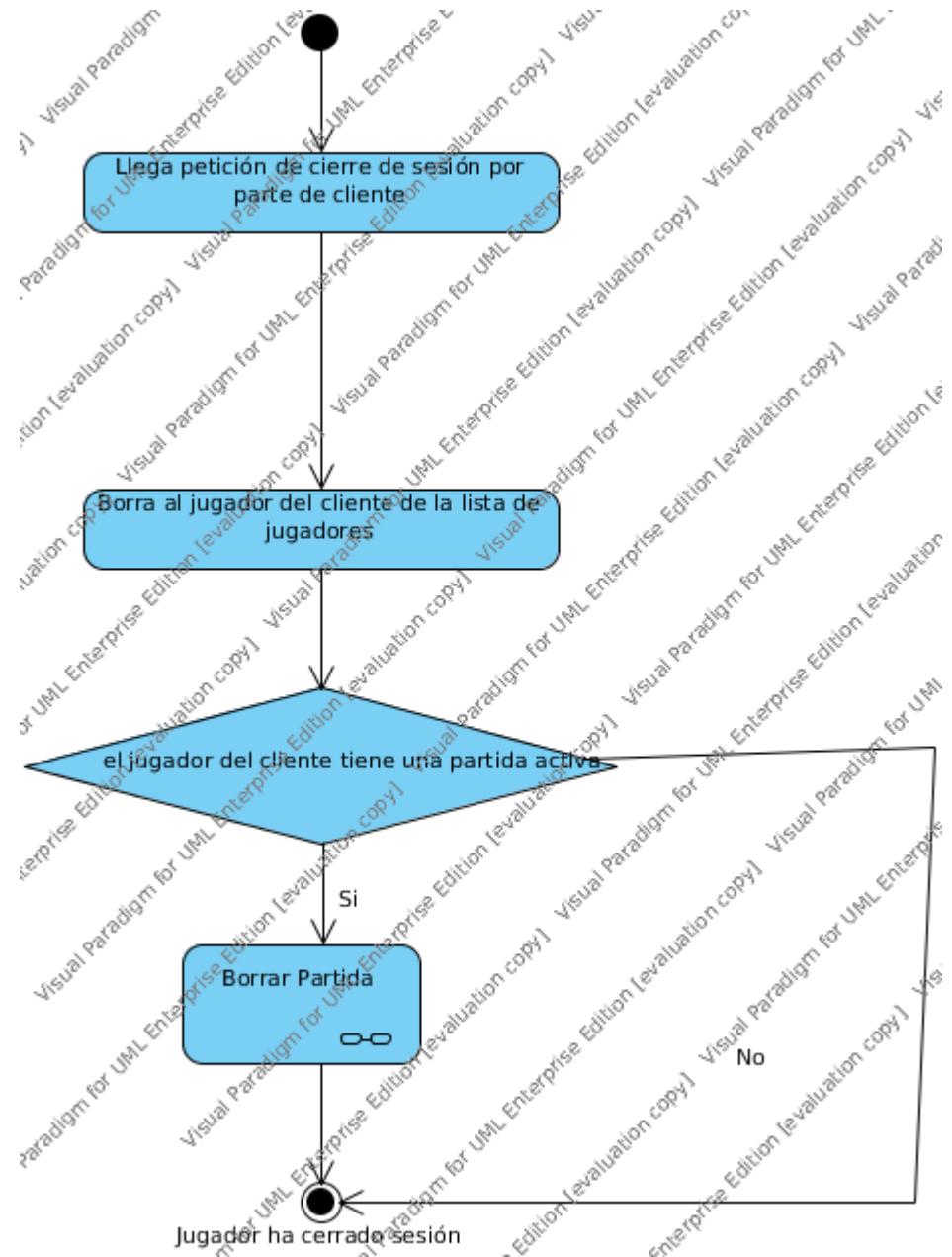
#### 6.2.1. Autenticar



#### 6.2.2. Borrar partida



### 6.2.3. Cerrar sesión

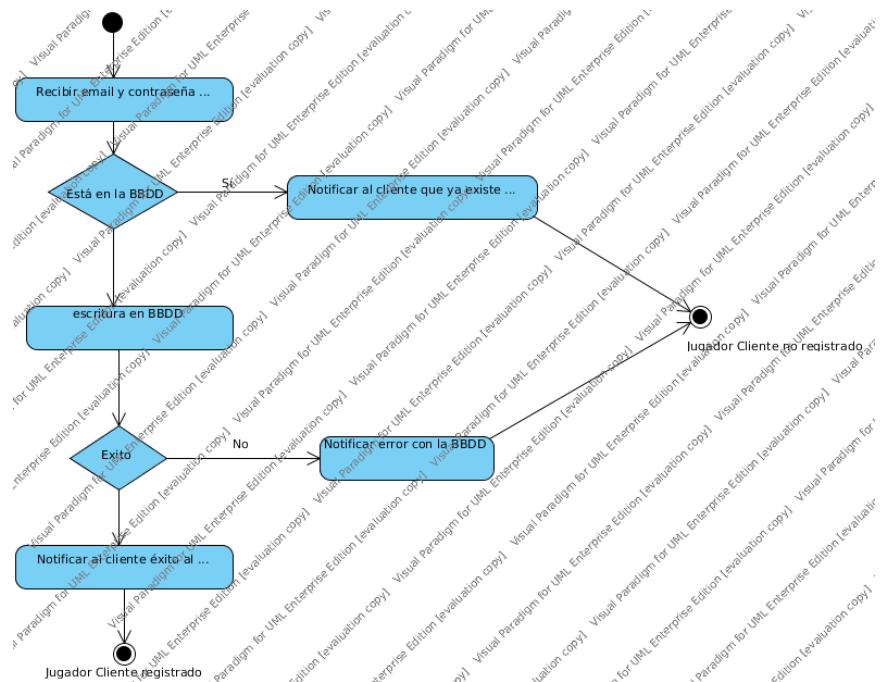


## 6 Máquinas de estado

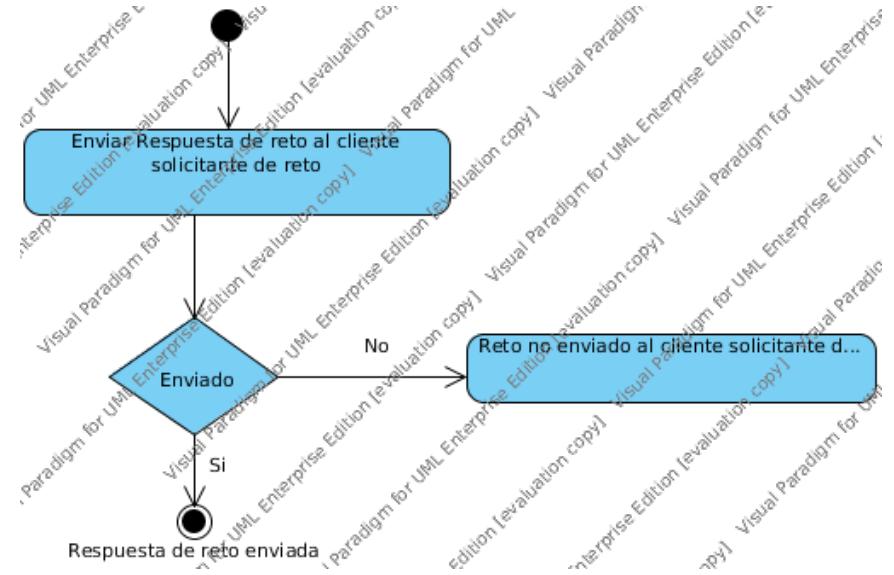
### 6.2.4. Poner ficha



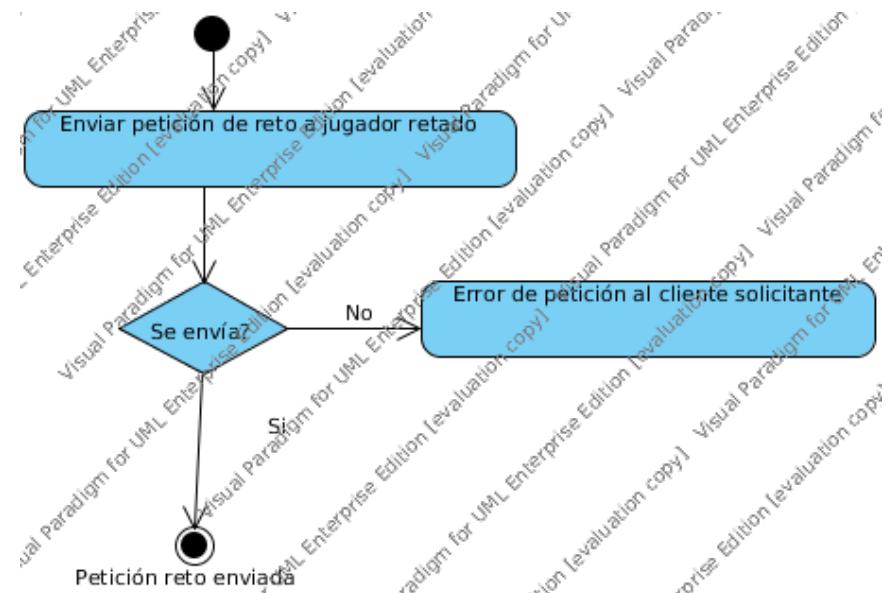
### 6.2.5. Registrar usuario



### 6.2.6. Respuesta de reto



### 6.2.7. Retar jugador



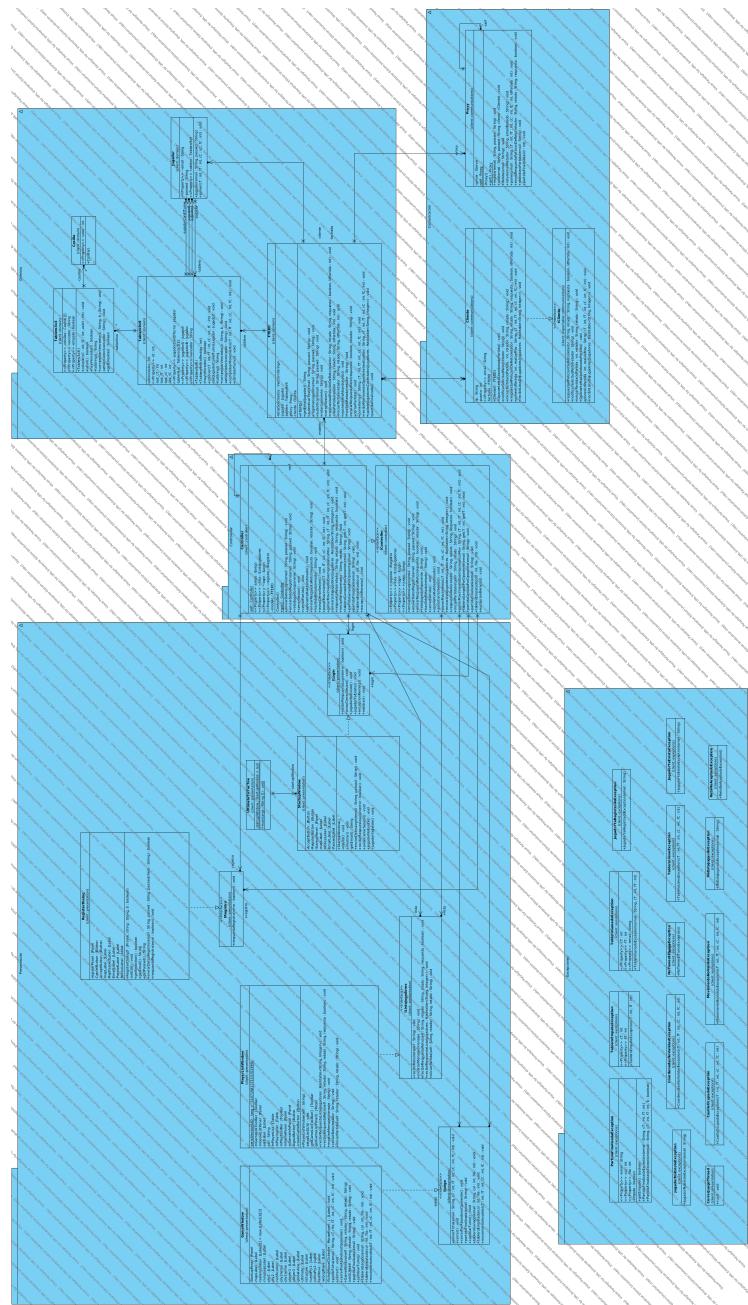
## 7. Diagrama de clases

### 7.1. Diagrama de clases del cliente

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

## 7 Diagrama de clases

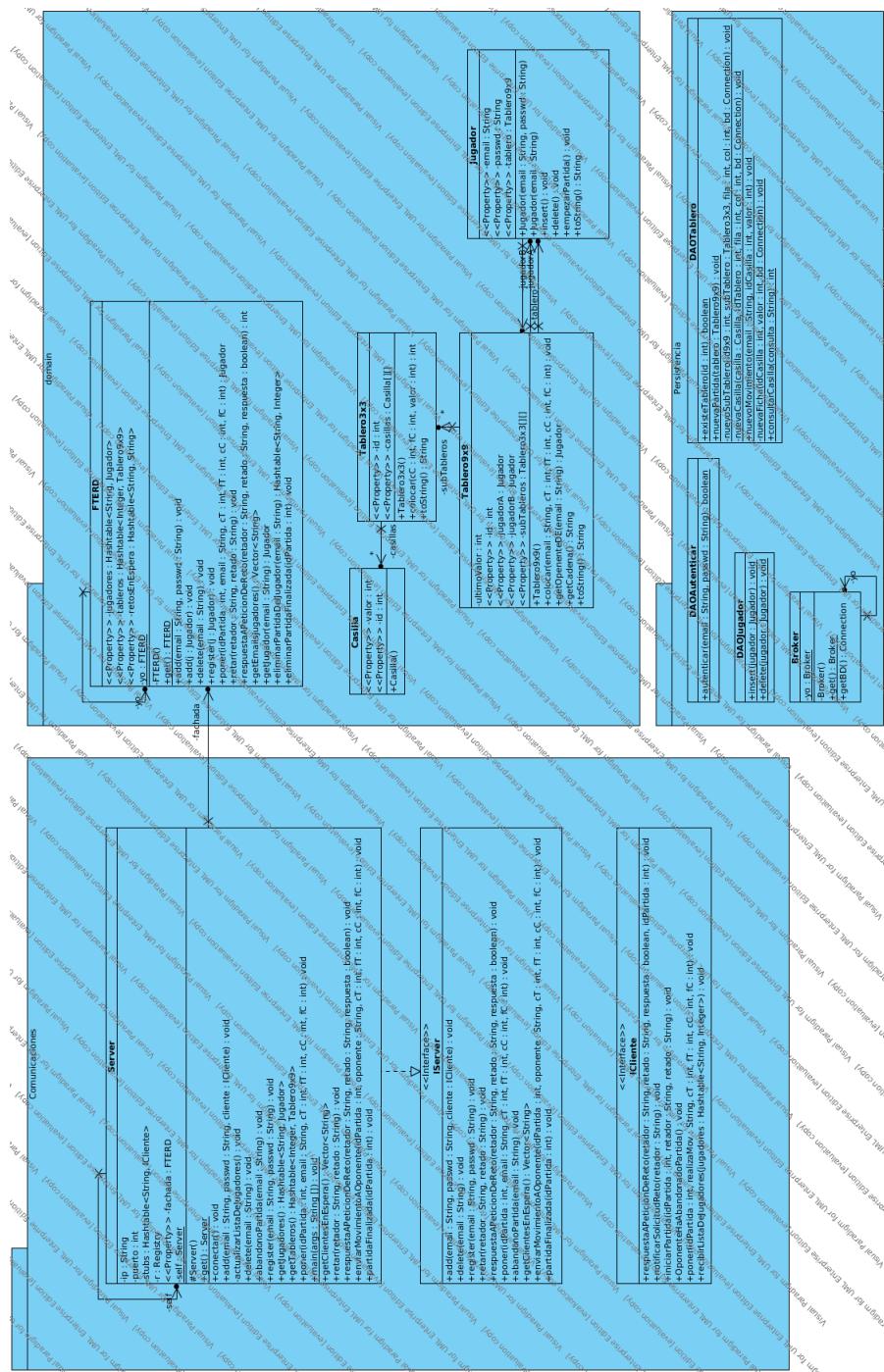
---



## 7.2. Diagrama de clases del servidor

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

## 7 Diagrama de clases



## 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 `tests.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

Los tests 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)

- Paquete `src.server.tests.communications`

Caso de uso	Test
Cerrar sesión	TestCerrarSesion
Registrar Usuario	TestRegistrarYAnyadirJugador
Autenticar	TestRegistrarYAnyadirJugador
Enviar respuesta a reto	TestRetoRechazar
Enviar respuesta a reto	TestRetoAceptar