



UNIVERSIDADE DA CORUÑA
FACULTADE DE INFORMÁTICA
Departamento de Computación

**PROXECTO DE FIN DE CARREIRA DE ENXEÑERÍA
TÉCNICA EN INFORMATICA DE XESTIÓN**

**DESENVOLVEMENTO DUNHA
APLICACIÓN QUE PERMITA XOGAR, EN
TEMPO REAL, POR INTERNET AO TUTE**

Autor: Paulino Villar Martínez
Director: José María García-Tizón Iglesias
Data: A Coruña, 23 de Septiembre de 2009

Agradecimientos

Quiero agradecer en primer lugar a mis padres por darme la oportunidad de estudiar esta carrera, y a mi hermana, por el apoyo que he recibido por su parte.

También quiero agradecer a mi tutor, José María García-Tizón Iglesias, por sus consejos, ayuda recibida y sus correcciones cuando me hicieron falta.

Y por último a mis amigos por apoyarme, animarme y darme esperanzas en todo momento.

Resumen

Hoy en día la gran mayoría de las personas que navegan por la red buscan diversión y entretenimiento, además de la búsqueda de información e intercambios de archivos. Cada año el número de personas con acceso a internet va en aumento, de igual manera aumenta el rango de edades en las que están comprendidos estos usuarios, con lo cual refleja que el uso de internet hoy en día ya no es sólo del mundo empresarial, sino que también incluye el ocio.

Por eso, este proyecto va centrado en la búsqueda de esa diversión y entretenimiento, centrándola en el juego de cartas conocido sobre todo a nivel nacional, llamado "Tute".

Va a consistir en la creación de una aplicación cliente que los usuarios podrán descargar de una web y ejecutar en sus computadoras sin ninguna dificultad, ya que está implementado de la forma más portable posible. Estos usuarios podrán jugar partidas de Tute por parejas o formar parte del público de una partida, todo esto mientras pueden hablar con el resto de usuarios, tanto por chat privado, general, o en la propia mesa. Los usuarios también irán luchando por ascender en un ranking de jugadores o podrán ver sus estadísticas privadas. Eso sí, para que estos usuarios puedan acceder a estas funcionalidades, deberán registrarse la primera vez que quieran entrar, creando su cuenta de jugador y después tendrán que autenticarse debidamente. Un usuario solo podrá acceder a su cuenta a través de una

sola sesión es decir, no pueden existir dos aplicaciones clientes conectadas utilizando la misma cuenta.

El proyecto incluye también la creación de un servidor, el cual controlará el juego de múltiples partidas simultáneamente, la conexión y desconexión de usuarios, los diferentes chats, a la vez que las diferentes consultas que se hagan para la obtención de las estadísticas. También guardará de forma permanente los datos importantes de una partida como son los jugadores, fecha y hora de inicio, mesa donde fue realizada y la puntuación de ésta.

Palabras clave

- Cliente
- Servidor
- Base de datos
- Java
- JDBC
- NetBeans
- UML
- MagicDraw UML
- MySql

Índice

1.	Introducción	- 10 -
1.1.	Contextualización	- 11 -
1.2.	Objetivo del proyecto	- 11 -
1.3.	Estructura de la memoria.....	- 13 -
2.	Fundamentos Tecnológicos	- 15 -
2.1.	Tecnología.....	- 16 -
2.2.	Lenguajes	- 16 -
2.3.	Herramientas	- 17 -
3.	Metodología Utilizada	- 18 -
3.1.	Elección de la metodología	- 19 -
3.2.	Aplicación de la metodología.....	- 19 -
4.	Análisis	- 21 -
4.1.	Casos de Uso	- 22 -
4.1.1.	Modelo de Casos de Uso.....	- 22 -
4.1.2.	Descripción de los actores.....	- 23 -
4.1.3.	Descripción de los casos de uso	- 24 -
4.2.	Diagramas de secuencia de sistema.....	- 31 -
4.3.	Modelo de clases conceptuales.....	- 37 -
4.3.1.	Diagrama de clases conceptuales	- 37 -
4.3.2.	Descripción de las clases conceptuales	- 39 -
5.	Diseño	- 42 -
5.1.	Diagramas de secuencia de bajo nivel.....	- 43 -
5.2.	Modelo clases software	- 53 -

5.3.	Diagrama de despliegue	- 55 -
5.4.	Modelo Relacional de la base de datos	- 57 -
6.	Implementación.....	- 58 -
6.1.	Primer incremento	- 59 -
6.2.	Segundo incremento	- 59 -
6.3.	Tercer incremento.....	- 60 -
6.4.	Cuarto incremento	- 61 -
6.5.	Quinto incremento	- 61 -
6.6.	Sexto incremento	- 62 -
6.7.	Séptimo incremento.....	- 62 -
6.8.	Octavo incremento.....	- 63 -
6.9.	Noveno incremento.....	- 64 -
7.	Pruebas	- 65 -
7.1.	Pruebas de unidad.....	- 66 -
7.2.	Pruebas de integridad.....	- 66 -
8.	Manual de Usuario	- 67 -
8.1.	Autenticarse	- 68 -
8.2.	Registrarse	- 70 -
8.3.	Iniciada la sesión	- 72 -
8.4.	Ver reglas.....	- 73 -
8.5.	V ranking	- 75 -
8.6.	Ver mesas	- 76 -
8.7.	Abrir mesa	- 78 -
8.8.	Sentado en mesa	- 80 -
8.9.	Fin de la partida	- 85 -
8.10.	Entrar como observador	- 85 -
8.11.	Barra de tareas	- 86 -

9.	Instalación del software	- 89 -
9.1.	Instalación de la maquina virtual de java	- 90 -
9.2.	Instalación del MySql.....	- 90 -
10.	Conclusiones y líneas futuras	- 92 -
11.	Bibliografía y enlaces	- 94 -
11.1.	Bibliografía.....	- 94 -
11.2.	Enlaces de consulta y descargas.....	- 94 -
12.	Apéndices.....	- 95 -
12.1.	Implementación física de la base de datos	- 96 -
12.2.	Algoritmo SHA-1	- 99 -
12.3.	Tecnologías empleadas.....	- 100 -
12.4.	Herramientas utilizadas.....	- 100 -
11.4.1.	Plataforma Java.....	- 100 -
11.4.2.	NetBeans	- 101 -
11.4.4.	MySql.....	- 104 -
11.4.3.	Otras herramientas:	- 106 -
12.5.	Lenguajes empleados	- 107 -
12.6.	Proceso unificado (RUP)	- 111 -
12.7.	Patrón MVC (modelo-vista-controlador).....	- 114 -

Índice de figuras

1.	Modelo Casos de Uso	22
2.	Diagrama de secuencia de sistema: abrir mesa	32
3.	Diagrama de secuencia de sistema: sentarse en mesa	33
4.	Diagrama de secuencia de sistema: observar mesa	34
5.	Diagrama de secuencia de sistema: Elegir Jugada	35
6.	Diagrama de secuencia de sistema: Realizar Cante	36
7.	Diagrama de clases conceptuales	38
8.	Diagrama de secuencia: Autenticarse	44
9.	Diagrama de secuencia: Abrir Mesa	45
10.	Diagrama de secuencia: Sentarse en mesa	46
11.	Diagrama de secuencia: Repartir Cartas	47
12.	Diagrama de secuencia: Realizar cante	48
13.	Diagrama de secuencia: Elegir Jugada	49
14.	Diagrama de secuencia: Carta ganadora	50
15.	Diagrama de secuencia: Sumar puntos baza	51
16.	Diagrama de secuencia: Calcular resultado	52
17.	Diagrama de clases software	54
18.	Diagrama de despliegue	56
19.	Modelo relacional	57
20.	Ventana: Autenticarse	68
21.	Ventana: Autenticarse "Error"	69
22.	Ventana: Registrarse	71
23.	Ventana: Registrarse "Error"	71
24.	Ventana: Registrarse "Confirmación"	72
25.	Ventana: Bienvenida	72
26.	Ventana: Reglas	73
27.	Ventana: Ver ranking	75
28.	Ventana: Ver mesas	76
29.	Tabla de mesas	77
30.	Ventana: Abrir mesa	78
31.	Menú abrir mesa	78
32.	Ventana: Invitación	79

33.	Menú invitación	80
34.	Ventana: Mesa	80
35.	Ventana: Partida	81
36.	Ventana: Partida en juego	82
37.	Menú jugador	82
38.	Centro de una mesa	83
39.	Menú estadística de una partida	84
40.	Ventana: Fin partida	85
41.	Ventana: Observador	86
42.	Barra de tareas	86
43.	Barra de tareas: Archivo	87
44.	Barra de tareas: Funciones	88
45.	Barra de tareas: Conexión	88
46.	NetBeans v6.7	102
47.	MagicDraw	103
48.	MySql	105

1. Introducción

En este apartado vamos a explicar los objetivos marcados en este proyecto así como la motivación y en el contexto donde nos encontramos.

1.1. Contextualización

Cada año que pasa, el número de usuarios que utiliza internet crece de forma exponencial, abarcando a más variedad de gente, que busca diferentes funcionalidades en su uso.

Muchas de estas personas, aunque no sea su uso habitual para la mayoría, buscan formas de entretenimiento y ocio a través de juegos, páginas de contacto, páginas de información de algún hobby, etc.

El número de lugares de este tipo, donde un usuario puede navegar disfrutando mientras lo hace, es enorme. En especial, cantidad de lugares ofrecen juegos en tiempo real, es decir, varios usuarios de distintas partes del mundo están realizando una partida en conjunto al mismo tiempo, ofreciendo a la vez que diversión, la posibilidad de relacionarse con personas de diferentes lugares.

1.2. Objetivo del proyecto

Este proyecto tiene como objetivo el desarrollo de una aplicación cliente/servidor, que haga la función descrita en el apartado anterior, de un juego cartas en tiempo real llamado Tute.

El proyecto, para explicarlo, vamos a dividirlo en tres, una parte que consistirá en el cliente, una segunda que estará formada por el servidor y, por último, una tercera que estará formada por una página web de la cual se podrán descargar los usuarios la parte cliente.

Comenzaremos explicando la aplicación cliente que es la parte dirigida al mayor número de personas. En esta aplicación es donde los nuevos usuarios podrán realizar su registro para poder autenticarse y disfrutar del conjunto del proyecto. Una vez autenticado estará situado en una sala donde podrá elegir asiento en diferentes mesas de juego para participar en una partida de "Tute por parejas", o si prefiere, situarse como público de una. Ambas cosas están acompañadas de diferentes chats (privados, de mesa y generales o de lobby) donde estos pueden participar en diferentes conversaciones. También podrán ver su situación en el ranking, ya que irán ganando o perdiendo puntos según su resultado en las partidas, o podrán ver sus estadísticas privadas, es decir, el resultado de sus partidas, con qué compañeros ha jugado cada una, las bazas que se realizaron en ellas y los contrincantes. Además en esta parte será posible ver las reglas del juego de "Tute por parejas".

La parte del servidor es la encargada de monitorizar el control de varias partidas de forma simultánea, de igual forma, también se ocupa de los chats y es el encargado de guardar los datos, tanto de los usuarios como de las partidas jugadas. Además también se encargará de realizar las consultas referentes a las diferentes estadísticas que pidan los usuarios conectados a través de una aplicación cliente.

Por último, la página web consistirá en un lugar donde los usuarios se podrán descargar el cliente, ver las reglas del "tute por parejas" además de poder autenticarse y acceder al ranking y sus estadísticas.

1.3. Estructura de la memoria

- **Introducción:** En este apartado se abre una puerta a la explicación del problema, y se exponen las motivaciones y objetivos que llevan a realizar el proyecto.
- **Fundamentos tecnológicos:** Se nombran tantos las tecnologías, lenguajes y herramientas utilizadas en el desarrollo del proyecto, dando las razones de su elección.
- **Metodología:** Apartado en el cual dedicamos tiempo a la explicación de la metodología utilizada, y el por qué de su elección.
- **Análisis:** Consiste en la exposición del análisis realizado del problema acompañado de los respectivos diagramas de UML realizados para su elaboración.
- **Diseño:** A la hora de crear un proyecto, el diseño sería el siguiente paso a seguir, por tanto, este apartado trata su explicación a través de los diagramas realizados para esta fase y su descripción.
- **Implementación:** Breve comentario de la implementación desarrollada, dividiéndola en los diferentes incrementos realizados. Cada uno de estos incrementos se definirá lo conseguido después del respectivo análisis, diseño y implementación.
- **Pruebas:** Explicación de las pruebas realizadas a la hora de probar el funcionamiento del proyecto.
- **Manual de usuario:** Se explica de forma breve y sencilla las funcionalidades a nivel usuario de la aplicación cliente.
- **Instalación del software:** Explica todo lo necesario para ejecutar tanto la aplicación cliente como la aplicación servidor en los diferentes ordenadores.

- **Conclusiones y líneas futuras:** Se hace una reflexión de los objetivos marcados comparándolos con el resultado final, además de añadir posibles mejoras a realizar.
- **Bibliografía:** Consiste en una enumeración tanto de los libros consultados como de las páginas webs.
- **Apéndices:** Contendrá una breve explicación de partes importantes correspondientes del proyecto, pero que se considera que no deben entrar en ningún punto de lo que se considera el cuerpo de la memoria.

2. Fundamentos Tecnológicos

A continuación se hará una enumeración de la tecnología empleada y el porqué de su elección.

2.1. Tecnología

La aplicación que vamos a crear debe cumplir con unos parámetros, comunes a la mayoría de las aplicaciones utilizadas en internet, que son los siguientes:

- Debe ser multiplataforma: Esto se refiere a que la aplicación debe correr en el mayor número de ordenadores distintos.
- Permitir acceso a base de datos: La aplicación debe obtener y guardar los datos de forma permanente.
- Poder integrarse con otras aplicaciones desarrolladas con diferentes tecnologías (en este caso para líneas futuras).
- Poseer una arquitectura multicapa.
- En especial para esta aplicación buscamos que sea software libre para llegar al mayor número posibles de personas, sin necesidad de realizar ningún tipo coste.

Por todas estas razones he decidido utilizar la plataforma de “Java”, ya que con sus características (explicadas en el apéndice correspondiente) permite la creación de una aplicación que cumpla todos estos parámetros.

Además ya es una tecnología muy extendida entre los internautas, ya que muchas de las páginas web más visitadas necesitan el aporte de esta plataforma para su perfecta visualización.

2.2. Lenguajes

Los lenguajes que podemos utilizar y que corran en la plataforma elegida son varios, en concreto todos aquellos sea compilados en “Bytecode”.

El más popular es el lenguaje “Java”, un lenguaje orientado a objetos y que cumple las características buscadas como que sea libre, y en el que dispondremos de diferentes “Apis” que podrán ser utilizadas.

Las características, así como un poco la historia de “Java” será explicada en su correspondiente apéndice además de algún otro lenguaje utilizado en menor proporción.

2.3. Herramientas

Se van a exponer las herramientas utilizadas en el desarrollo del proyecto para dar una visión global de los recursos utilizados.

En su elección siempre se trató de buscar el buen funcionamiento y la aportación de funcionalidades que ayudarán a la creación del proyecto.

Las herramientas utilizadas fueron:

- MagicDraw UML 16.0.
- NetBeans IDE 6.7.
- MySql 5.4.1 versión beta.
- Adobe Photoshop.

3. Metodología Utilizada

En el contenido de este apartado voy a explicar que metodología he elegido para el desarrollo del proyecto y sus razones.

3.1. Elección de la metodología

En el desarrollo del software es importante elegir una metodología que nos marque los pasos del proceso y que lo haga de forma estructurada y eficiente.

En este proyecto hemos elegido seguir una metodología llamada Proceso Unificado o RUP. Es una metodología que caracterizada por:

- Ser dirigida por los casos de uso.
- Centrado en la arquitectura.
- Realización de enfoque de riesgos.
- Iterativo e incremental.

A parte de por sus características, la decisión vino dada por tratarse de una marca de trabajo extensible que puede ser adaptado a proyectos específicos.

La metodología será explicada en su correspondiente apéndice.

3.2. Aplicación de la metodología

Una vez elegida la metodología a seguir, comenzamos siguiendo ésta paso a paso. Primero, la "Fase Inicio", donde justificamos la creación del proyecto y donde marcamos los objetivos a cumplir, a la vez que realizamos la detección de los diferentes casos de uso.

A continuación, en la "Fase de Elaboración" a partir de los casos de uso realizamos la arquitectura del sistema, a partir de la detección de las diferentes clases, y marcando las interacciones entre ellas a partir de los diagramas de secuencia. También fue estudiada la conexión entre paquetes

necesaria para su construcción realizando el respectivo diagrama de paquetes.

Lo siguiente fue la "Fase de construcción", que consiste en una iteración centrada sobretodo en la implementación del sistema, abordándolo a través de pequeños incrementos, los que concluían con parte software ejecutable

La "Fase de transacción", que sería la última iteración a realizar en la metodología, consiste en el despliegue hacia los usuarios finales. Esta iteración no está marcada en los objetivos, aun así, el sistema fue desplegado a un número pequeño de usuarios para obtener los datos que fueron añadidos en su refinamiento.

4. Análisis

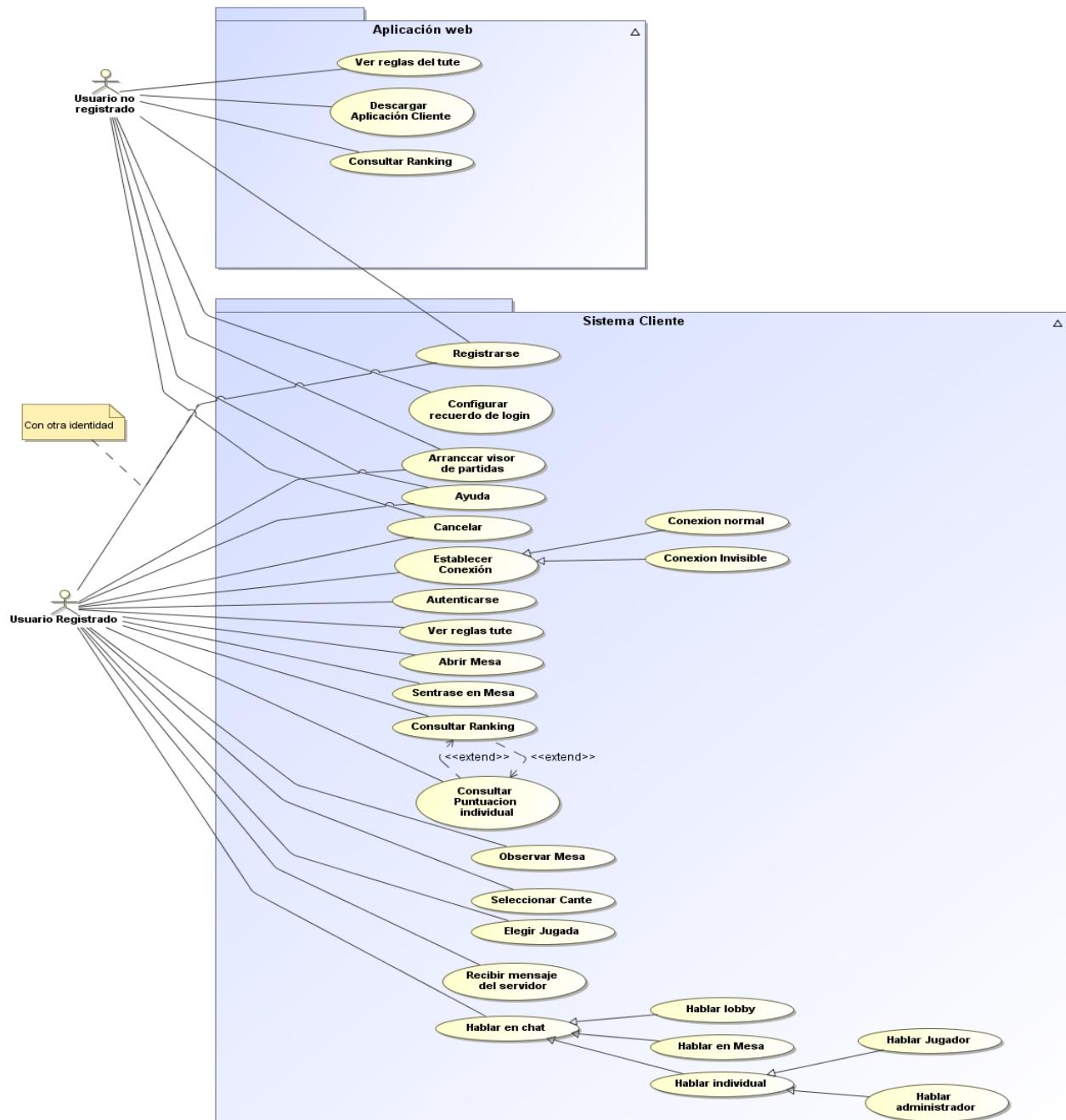
En este apartado se va a realizar una descripción del análisis realizado en el desarrollo del proyecto, mostrando los diferentes diagramas correspondientes.

4.1. Casos de Uso

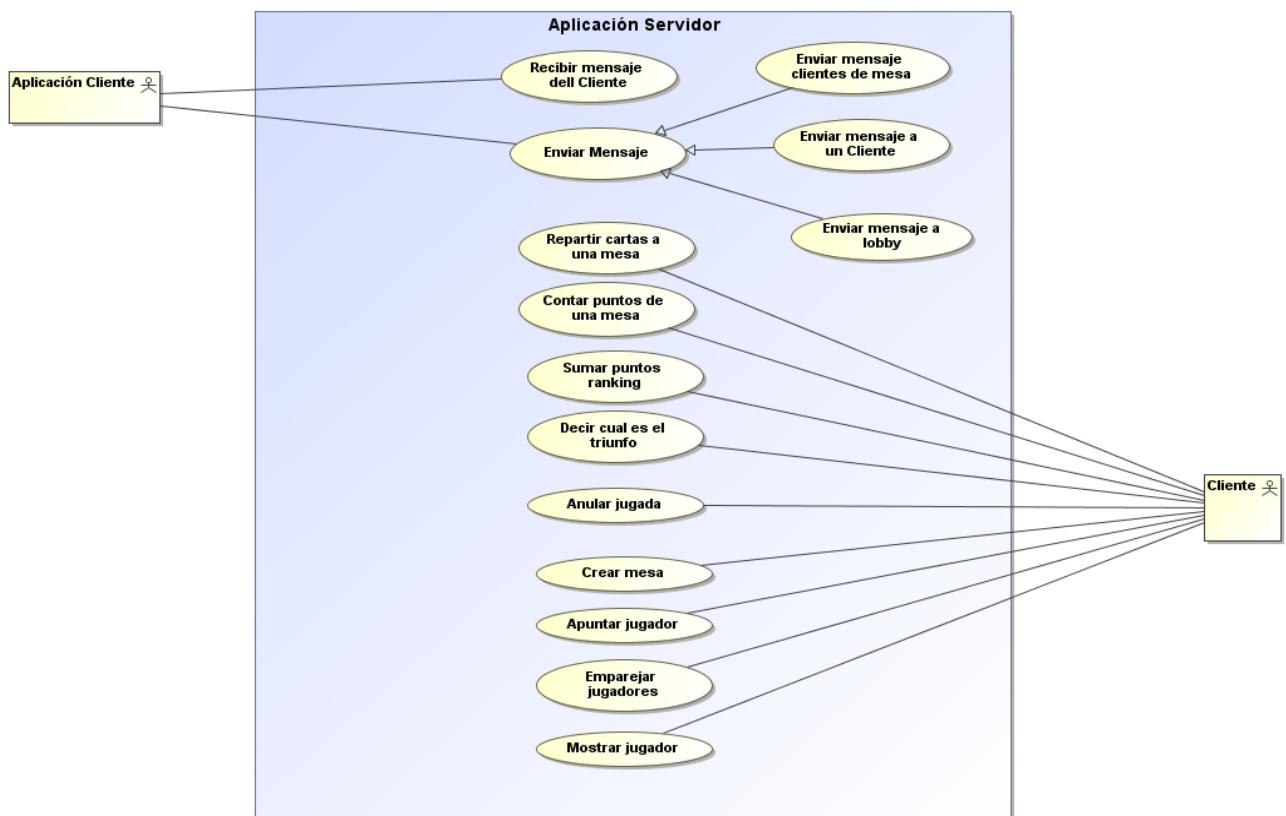
Lo principal de un análisis es buscar los requisitos del sistema, para eso es primordial buscar los casos de uso a realizar por los diferentes usuarios.

Primero, se mostrará el diagrama de casos de uso y a continuación una breve descripción de los primordiales.

4.1.1. Modelo de Casos de Uso



Desarrollo de una aplicación que permita jugar, en tiempo real, por internet al Tute



1 Modelo Casos de Uso

4.1.2. Descripción de los actores

En los casos de uso, se llama actor a una persona u otro sistema que intercambia información con nuestro sistema. Una misma persona física o un mismo sistema pueden estar representados por varios actores distintos ya que pueden jugar con varios roles a la hora de intercambiar información. Ahora, vamos a hacer una breve descripción de los actores que se encuentran en nuestro modelo.

Usuario no Registrado: Con este actor nos referimos a toda persona que no esté autenticada en el sistema.

Usuario Registrado: Nombre que le damos a aquella persona que está registrada y autenticada en el sistema.

Aplicación Cliente: Nos referimos a nuestro sistema cliente, que es quien envía información a sistema Servidor.

Cliente: Da nombre a las diferentes aplicaciones clientes que van a recibir información en los diferentes casos de uso.

4.1.3. Descripción de los casos de uso

Los casos de uso más importantes, o de más relevancia, van a ser descritos a continuación, diciéndose de cada uno el nombre, el actor principal, una breve descripción, el flujo de secuencia, las pre y pos condiciones, la frecuencia con la que sucede y un escenario alternativo en caso que haya posibilidad:

Autenticarse:

Nombre:	Caso de uso Autenticarse
Actor Principal:	Usuario Registrado
Descripción:	Un usuario quiere proceder a conectarse al servidor entonces debe iniciar su sesión introduciendo sus datos.
Secuencia:	<ol style="list-style-type: none">1. El usuario registrado introduce el nombre y contraseña,2. El sistema cliente envía los datos al sistema servidor3. El sistema servidor envía confirmación al sistema cliente4. El sistema cliente muestra la confirmación de

	contraseña y usuario válido. 5. El sistema muestra la sala de juego.
Pre Condición:	1. El usuario debe estar registrado
Frecuencia:	Muy Alta
Escenario Alternativo:	*a. El servidor se cierra 1. El sistema cliente muestra un mensaje de error en el servidor al usuario registrado 3.a. El usuario registrado introduce mal su contraseña o usuario 1. El sistema servidor devuelve un error al sistema cliente. 2. El sistema cliente muestra el error al usuario

Abrir mesa:

Nombre:	Caso de uso de abrir mesa
Actor Principal:	Usuario Registrado
Descripción:	Cuando un usuario registrado quiere crear una mesa, el sistema le pide unos datos de inicio como el nombre de jugadores, en caso de querer realizar una invitación.
Secuencia:	1. El usuario pulsa en “crear mesa”. 2. El sistema cliente envía petición al sistema servidor. 3. El sistema servidor envía confirmación de la petición

	<ol style="list-style-type: none">4. El sistema cliente pide nombre de jugadores al usuario registrado5. El usuario registrado introduce los nombres.6. El sistema cliente le pide las opciones al usuario registrado7. El usuario registrado introduce las opciones8. El sistema cliente pide al sistema servidor que cree la mesa con los datos.9. El sistema servidor devuelve la mesa al sistema cliente10. El sistema cliente muestra la mesa al usuario registrado
Pre condición:	<p>El usuario debe estar registrado</p> <p>El usuario debe estar autenticado</p>
Frecuencia:	Media
Escenario alternativo:	<p>*a. Se produce un error en el servidor</p> <ol style="list-style-type: none">1. El sistema cliente muestra un mensaje de error en el servidor a la aplicación cliente. <p>3.a. El sistema servidor está lleno</p> <ol style="list-style-type: none">1. El sistema servidor envía un mensaje al sistema cliente negándole la posibilidad.2. El sistema cliente muestra al usuario registrado un mensaje comunicándole opción no disponible. <p>9.a. El usuario registrado había introducido mal algún nombre, es decir el nombre de un usuario registrado</p>

	<p>que no existe.</p> <ol style="list-style-type: none">1. El sistema servidor envía un mensaje al sistema cliente diciendo nombres no válidos.2. El sistema cliente muestra un mensaje de error al usuario registrado comunicándole que los nombres no son válidos.3. El sistema cliente pide al usuario registrado que vuelva a introducir los nombres.
--	---

Sentarse en mesa

Nombre:	Caso de uso de sentarse en mesa
Actor principal:	Usuario Registrado
Descripción:	Un usuario registrado quiere ocupar un asiento de una mesa para realizar una partida.
Secuencia:	<ol style="list-style-type: none">1. El usuario registrado elige un asiento2. El sistema cliente envía la petición al sistema servidor3. El sistema servidor envía la información de la mesa al sistema cliente4. El sistema cliente muestra la mesa al usuario registrado <p>*En caso de que ya estén cuatro jugadores en la mesa</p>

	<ol style="list-style-type: none">5. El sistema servidor envía información de la mano a la aplicación cliente6. El sistema cliente muestra la mano al usuario registrado7. El sistema servidor envía información del triunfo al sistema cliente8. El sistema cliente muestra el triunfo al usuario registrado.9. El sistema servidor envía datos del repartidor al sistema cliente.10. El sistema cliente muestra el repartidor al usuario registrado
Pre condición:	<p>El usuario debe estar registrado</p> <p>El usuario debe estar autenticado</p> <p>El asiento debe estar marcado como libre.</p>
Frecuencia:	Muy alta
Escenario alternativo:	<p>*a. Si el servidor se cae en cualquier momento</p> <ol style="list-style-type: none">1. El sistema cliente muestra un mensaje de error en el servidor al usuario registrado3.a. El asiento está ocupado<ol style="list-style-type: none">1. El sistema servidor envía un mensaje al sistema cliente notificándole que el asiento está ocupado.2. El sistema cliente muestra un mensaje de error al usuario registrado

Observar mesa:

Nombre:	Caso de uso observar mesa
Actor Principal:	Usuario registrado
Descripción:	Un usuario registrado pide la opción de entrar como público a una partida
Secuencia:	<ol style="list-style-type: none">1. El usuario registrado selecciona entrar como publico en una mesa2. El sistema cliente envía la petición al sistema servidor3. El sistema servidor envía la mesa al sistema cliente4. El sistema cliente muestra la mesa al usuario registrado
Pre condición:	<ol style="list-style-type: none">1. El usuario debe estar registrado2. El usuario debe estar autenticado
Frecuencia:	Media
Escenario alternativo:	<p>*a. Si el servidor se cae en cualquier momento</p> <ol style="list-style-type: none">1. El sistema cliente muestra un mensaje de error en el servidor al usuario registrado. <p>3.a. La mesa no admite público</p> <ol style="list-style-type: none">1. El sistema servidor envía un mensaje notificándoselo al sistema cliente.2. El sistema cliente muestra un mensaje de información al usuario registrado

Elegir jugada:

Nombre:	Caso de uso elegir jugada
Actor principal:	Usuario registrado
Descripción:	Sucede cuando un usuario registrado elige una carta para jugarla en una baza.
Secuencia:	<ol style="list-style-type: none">1. El usuario registrado selecciona una carta para jugar.2. El sistema cliente envía la carta al sistema servidor3. El sistema servidor manda la confirmación al sistema cliente
Pre condición:	<ol style="list-style-type: none">1. El usuario debe estar registrado2. El usuario debe estar autenticado
Frecuencia:	Muy alta
Escenario alternativo:	<p>*a. Si el servidor se cae en cualquier momento</p> <ol style="list-style-type: none">1. El sistema cliente muestra un mensaje de error en el servidor al usuario registrado. <p>2.a. La carta no es válida</p> <ol style="list-style-type: none">1. El sistema cliente muestra un mensaje informativo al usuario registrado

Realizar cante:

Nombre:	Caso de uso realizar cante
Actor principal:	Usuario registrado
Descripción:	Un usuario registrado quiere realizar un cante cuando está jugando una partida.
Secuencia:	<ol style="list-style-type: none">1. El usuario registrado marca un cante2. El sistema cliente envía el cante al sistema servidor.3. El sistema servidor confirma la recepción al sistema cliente.
Pre condición:	<ol style="list-style-type: none">1. El usuario debe estar registrado2. El usuario debe estar autenticado
Frecuencia:	Muy alta
Escenario alternativo:	<p>*a. Si el servidor se cae en cualquier momento</p> <ol style="list-style-type: none">1. El sistema cliente muestra un mensaje de error en el servidor al usuario registrado. <p>2.a. El cante no es válido.</p> <ol style="list-style-type: none">1. El sistema cliente muestra un mensaje informativo al usuario registrado

4.2. Diagramas de secuencia de sistema

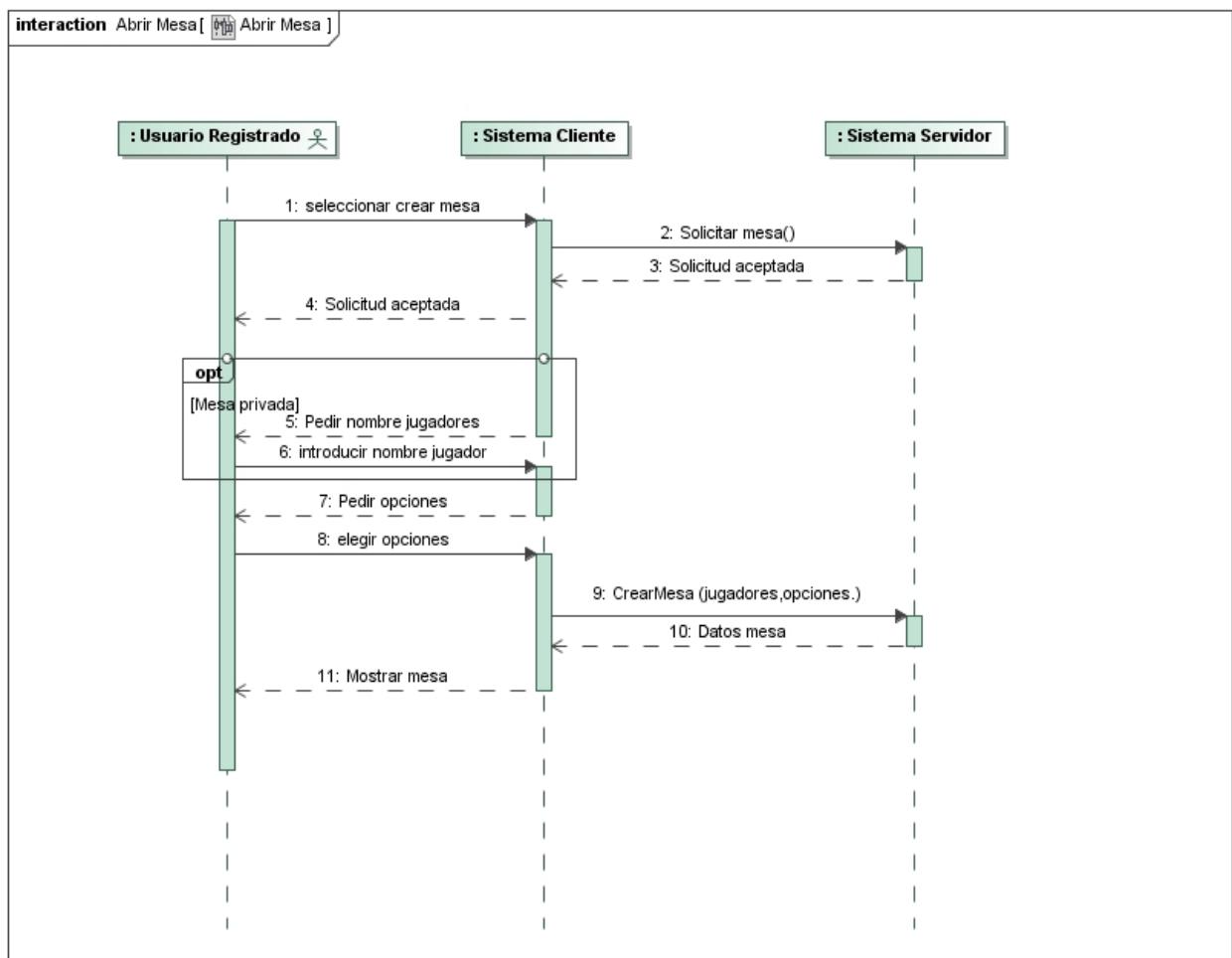
Los diagramas de secuencia de sistema son una forma fácil de mostrar los eventos de entrada y salida de un sistema. Cada diagrama de secuencia muestra un curso de eventos indicando a los actores que interactúan con el sistema, y al propio sistema como una caja negra. Cuando decimos que el

sistema se representa como una caja negra, es que no se muestra lo que sucede en el interior del sistema, simplemente se ven las entradas y salida que recibe y produce.

A continuación vamos a mostrar el diagrama de secuencia de los casos de usos más relevantes anteriormente citados.

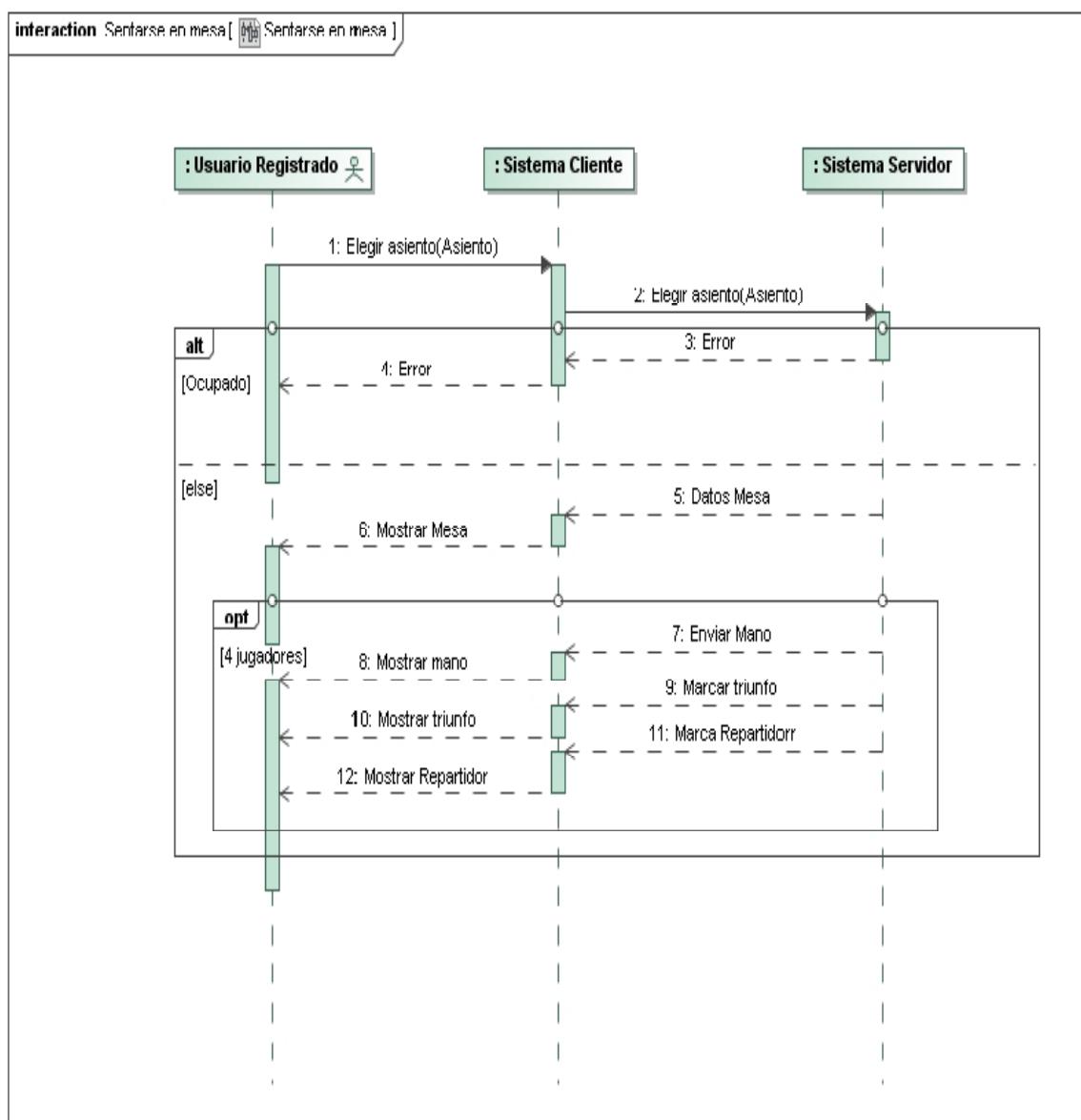
Abrir mesa

Representa el flujo de eventos que suceden cuando un usuario registrado abre una mesa nueva.



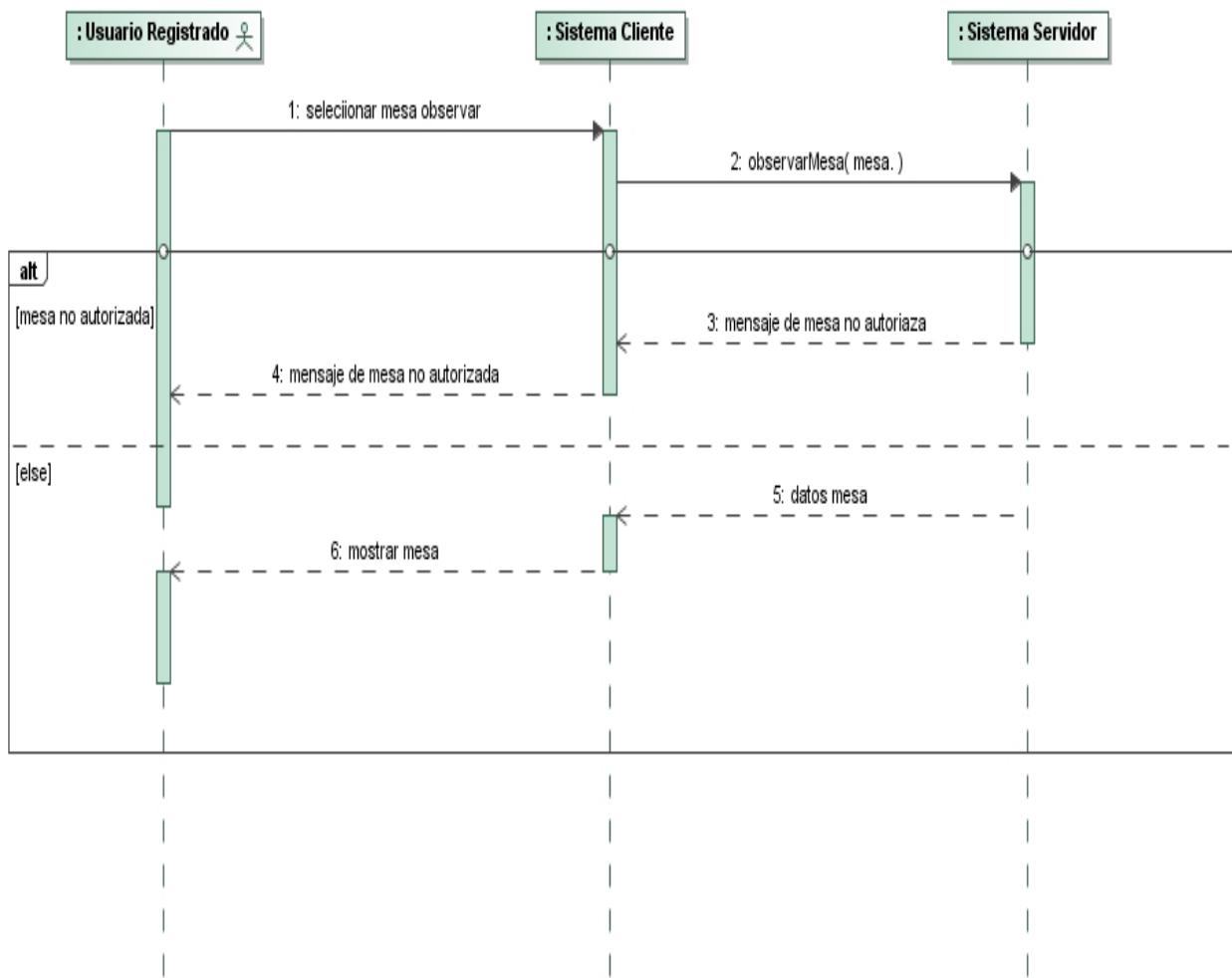
Sentarse en mesa:

Muestra la secuencia de eventos sucedidos cuando un usuario registrado selecciona un asiento para entrar en una mesa en concreto. Mostrando el posible flujo alternativo que puede suceder si la mesa está marcada como privada, es decir, que los asientos están reservados para unos determinados jugadores.



Observar mesa

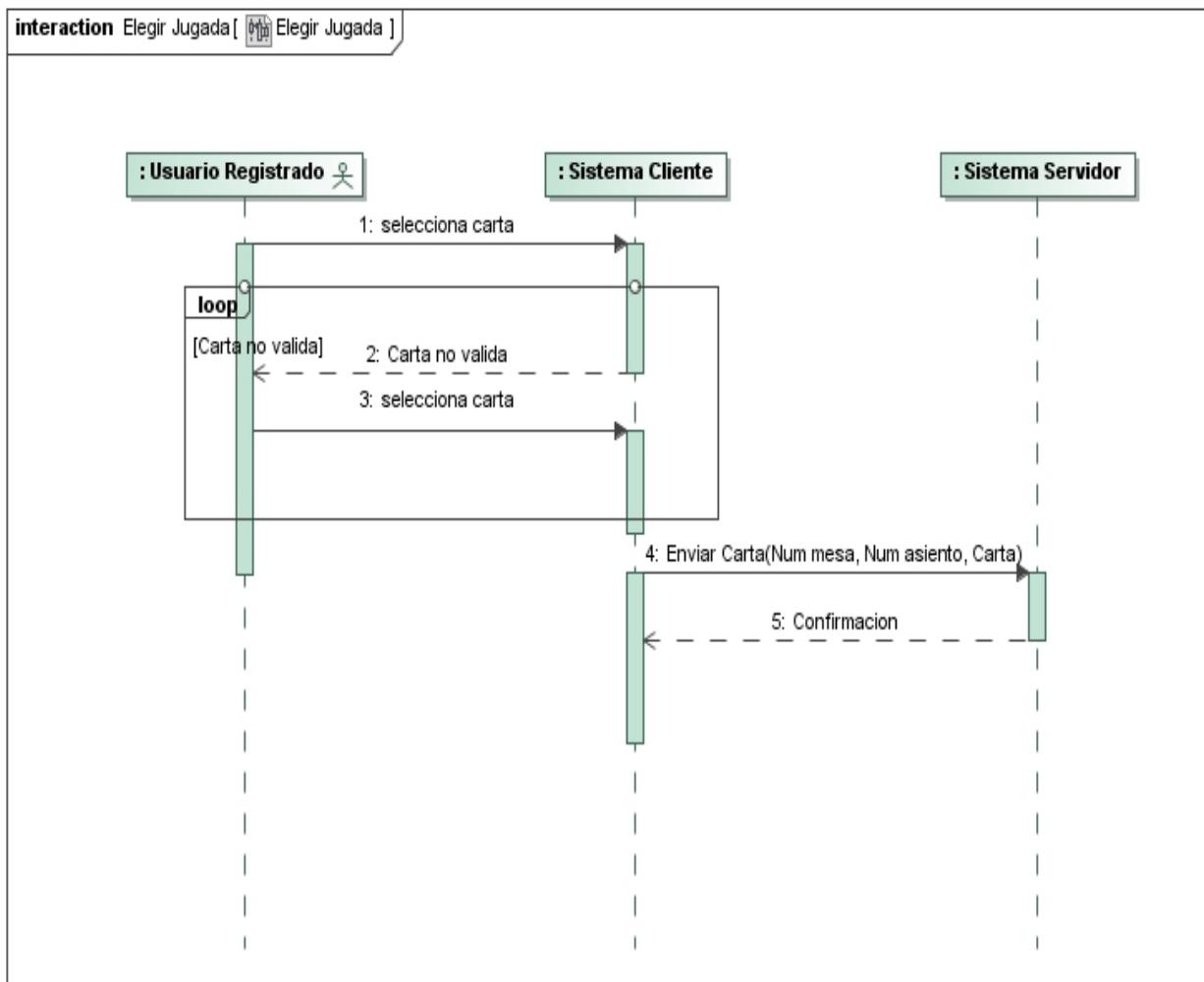
Se puede observar el flujo de eventos sucedidos cuando un usuario registrado selecciona para entrar a una mesa como público. El flujo alternativo es si la mesa no está autorizada al público es decir que nadie puede verla.



4Diagrama de secuencia de sistema: observar mesa

Elegir jugada

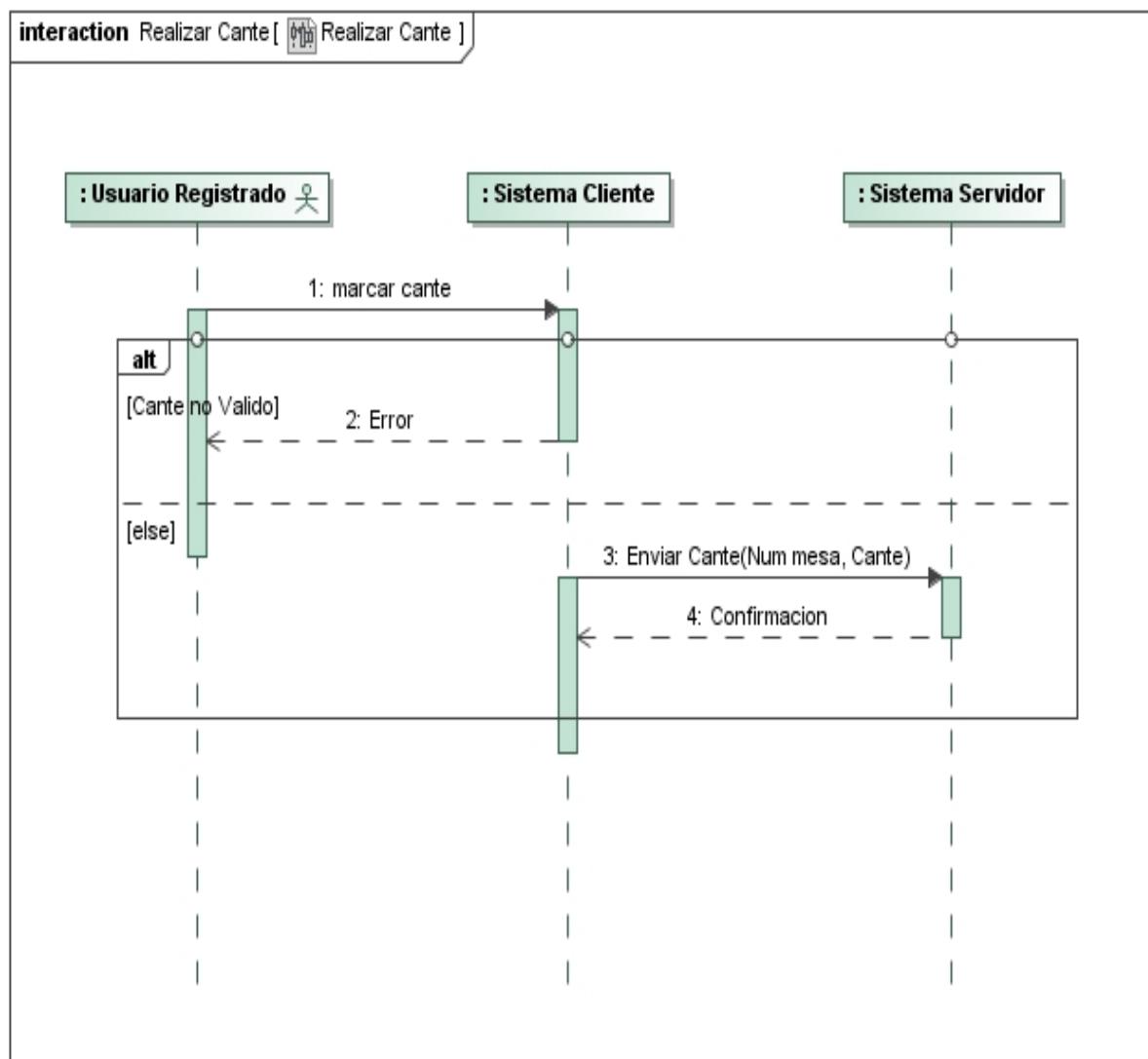
Consiste en representar la secuencia de eventos producidos cuando un usuario registrado elige una carta para tirar a la baza en juego. Podemos comprobar que el sistema cliente será el encargado si una carta es válida para la baza.



5Diagrama de secuencia de sistema: Elegir Jugada

Realizar cante:

Marca el flujo de eventos que suceden cuando un usuario registrado selecciona un cante a realizar. Igual que en el caso de uso de elegir jugada, el sistema cliente es el encargado de comprobar si el cante es válido.



6Diagrama de secuencia de sistema: Realizar Cante

4.3. Modelo de clases conceptuales

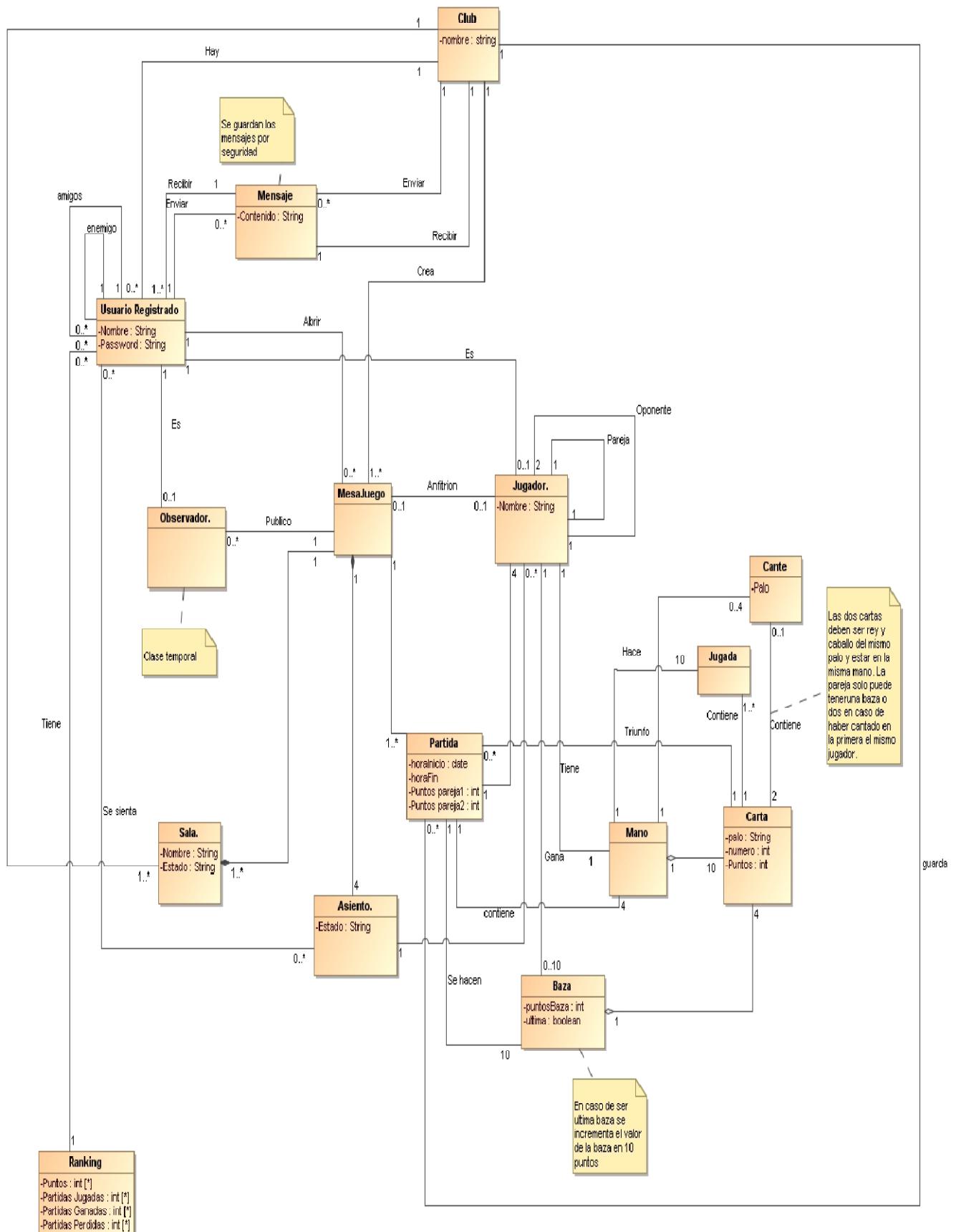
4.3.1. Diagrama de clases conceptuales

Un diagrama de clases es un tipo de diagrama estático encargado de definir la estructura del sistema, las clases que lo componen, con sus atributos, y las distintas relaciones entre ellas.

Los diagramas de clases conceptuales, en concreto, no definen clases software sino que dan una idea general sobre la información que deberá manejar el sistema según el análisis realizado hasta el momento.

El diagrama de clases conceptuales realizado en este análisis es el mostrado en la página siguiente.

Desarrollo de una aplicación que permita jugar, en tiempo real, por internet al Tute



4.3.2. Descripción de las clases conceptuales

Club: Todo conjunto de personas con mesas de juegos forman lo que se le suele denominar como “club”. Es decir, con esta clase nos referimos a los diferentes lugares donde estos usuarios van a jugar.

Mensajes: Se refiere a los mensajes que se envían por los diferentes chats, entre usuarios, a un chat general del club.

Usuario registrado: Toda persona que está registrada en el sistema

Observador: Da nombre al usuario registrado cuando entra como público a una partida. Entonces esta clase representa a todos aquellos usuarios que están situados en una mesa no para jugar, sino para ver.

Mesa juego: Lugar donde se van a realizar las partidas de juego.

Jugador: Cuando un usuario registrado entra en una mesa para jugar, se convierte en un jugador. Es decir, esta clase representa a todos los usuarios que están situados en una mesa jugando o preparados para jugar.

Cante: Representa los diferentes cantes que puede realizar un jugador en una partida. Los cantes pueden ser 20 en oros, 20 en copas, 20 en bastos, 20 en espadas o las 40.

Jugada: Se refiere a la acción que hace un jugador cuando lanza una carta, o elige una para ser jugada en una baza.

Carta: Clase referente al objeto físico de una carta.

Mano: Nos referimos al conjunto de cartas que son repartidas a principio de una partida a un jugador. Es decir, todas las cartas que le tocaron a un jugador, forman la mano que tendrá para la partida.

Partida: Cuando en una mesa de juego se encuentran ya cuatro jugadores, y se comienzan a repartir las cartas, es cuando se inicia una partida, entonces con esta clase nos referimos a todo lo que conlleva desde su inicio hasta cuando no quedan más cartas por jugar.

Baza: Las partidas se reparten en 10 ciclos, cada ciclo está formado por una carta que lanzó cada jugador, y cada uno de estos ciclos recibe el nombre de baza.

Asiento: Un jugador cuando está situado en una mesa de juego, tanto jugando como esperando que se inicie, está ocupando un lugar o posición en la mesa, que es a lo que nos referimos con asiento.

Sala: Dentro de un club de usuarios, se encuentra la sala donde se sitúan las mesas de juego, y donde los usuarios registrados entran tanto a jugar o observar una partida.

Ránking: Nos referimos a la clasificación de los usuarios según los puntos que van consiguiendo según el resultado de sus partidas.

5. Diseño

A continuación vamos a explicar el diseño correspondiente a este proyecto, mostrando los diagramas realizados y una explicación de cada uno de ellos.

5.1. Diagramas de secuencia de bajo nivel

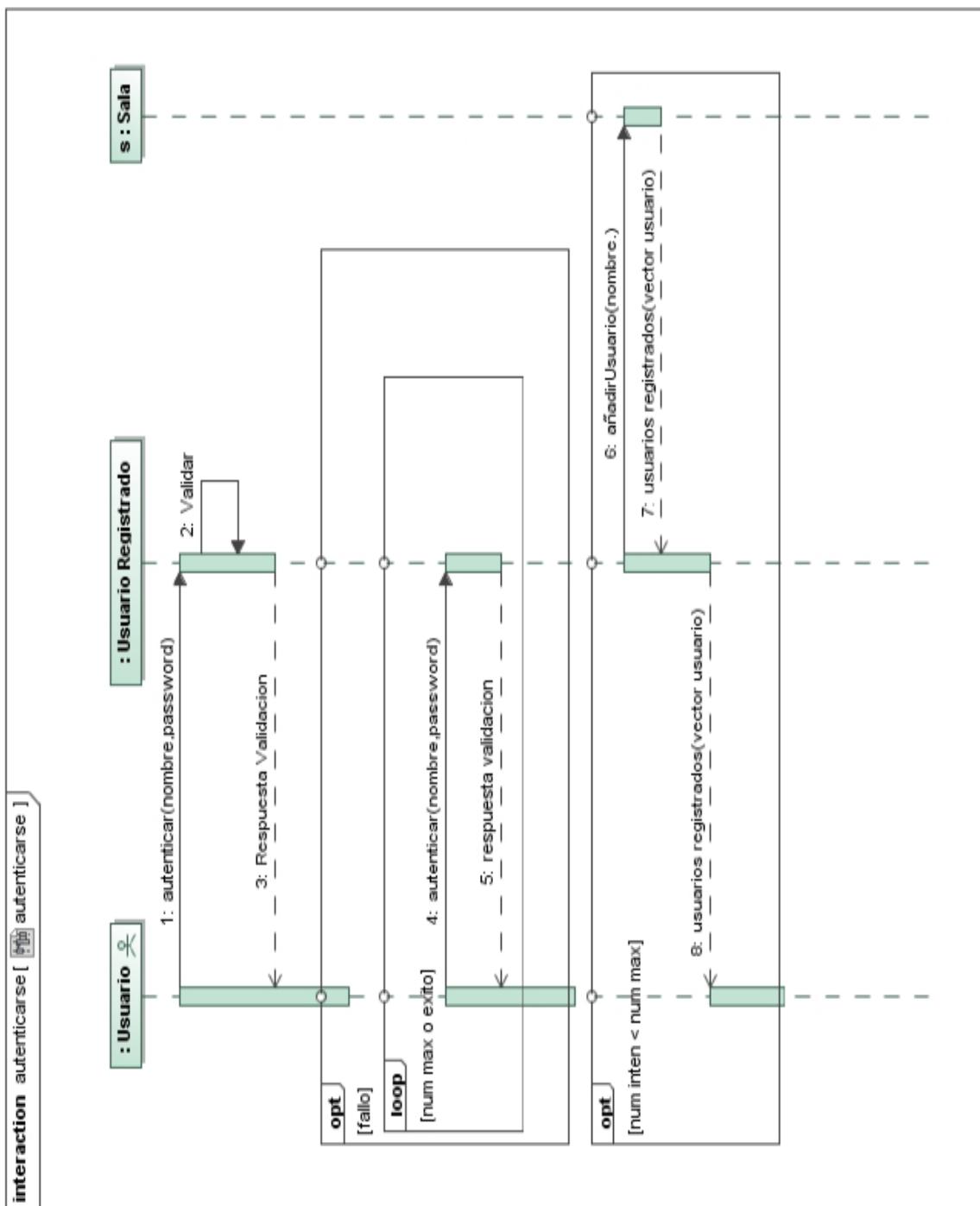
Los diagramas de secuencia de bajo nivel, igual que los diagramas de secuencia en el análisis, formaban el estudio dinámico; ahora en el diseño, son los diagramas de secuencia de bajo nivel.

Se llaman diagramas de secuencia de bajo nivel porque disminuyen el nivel de abstracción, es decir, con ellos vamos a poder ver el flujo de eventos que ocurre dentro del sistema al realizarse un caso de uso. Por este motivo, también se le llaman diagramas de secuencia de caja blanca.

A continuación, están los diagramas de secuencia de bajo nivel realizados para el diseño de este proyecto:

Autenticarse:

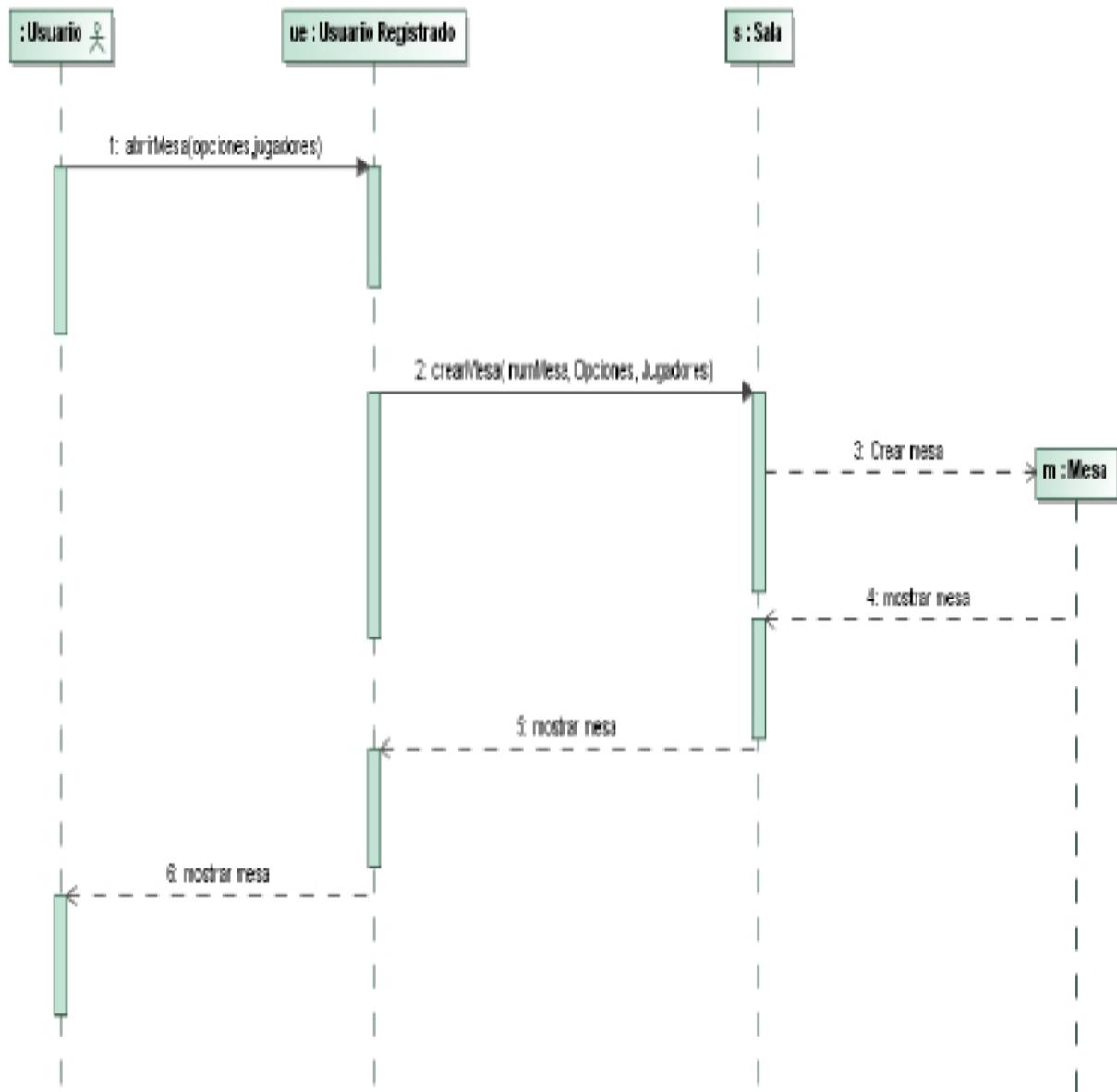
En él podemos observar lo que sucede dentro del sistema cuando un usuario registrado realiza el caso de uso de autenticarse, es decir, introduce sus datos de usuario para conectarse al servidor, viendo los posibles errores que pueden suceder .



11Diagrama de secuencia: Autenticarse

Abrir mesa:

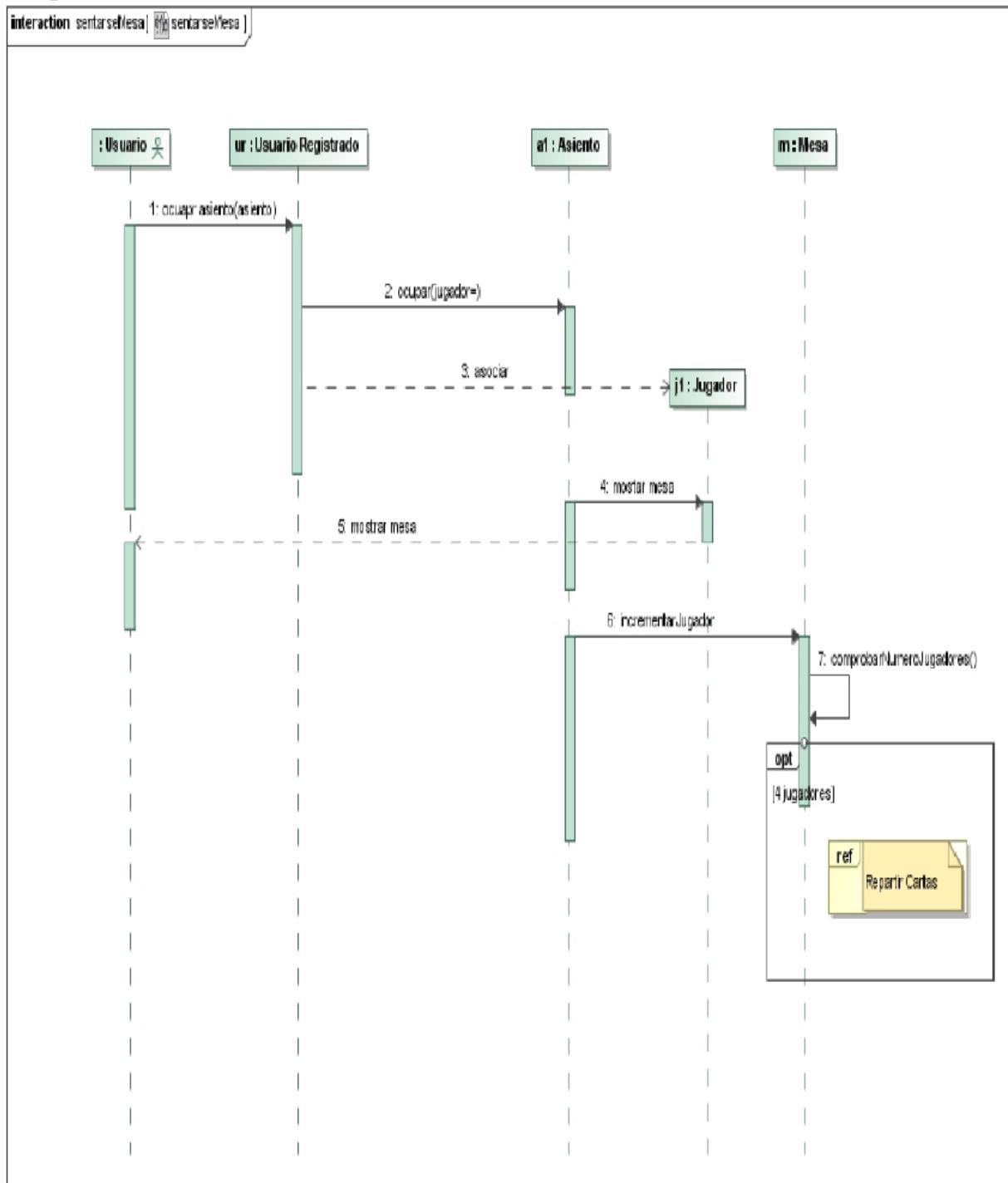
En él podemos observar lo que sucede dentro del sistema cuando un usuario registrado realiza el caso de uso abrir mesa.



12. Diagrama de secuencia: Abrir Mesa

Sentarse en mesa

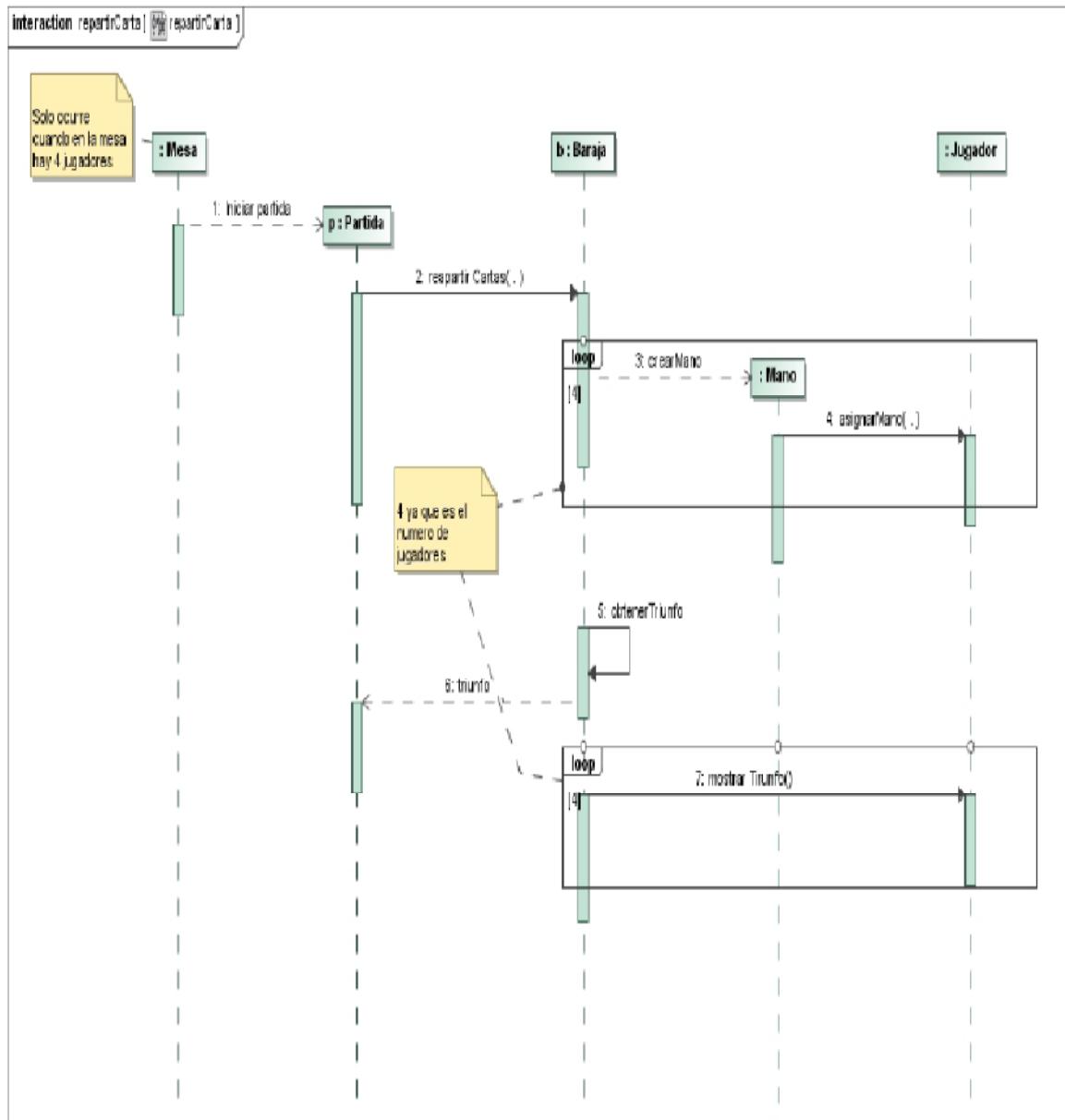
Muestra el flujo de eventos que suceden dentro del sistema cuando un usuario registrado quiere sentarse en una mesa. En él también podemos ver las futuras clases que van a ser las encargadas de controlar los distintos aspectos de la acción.



8 Diagrama de secuencia: Sentarse en mesa

Rearrange cards

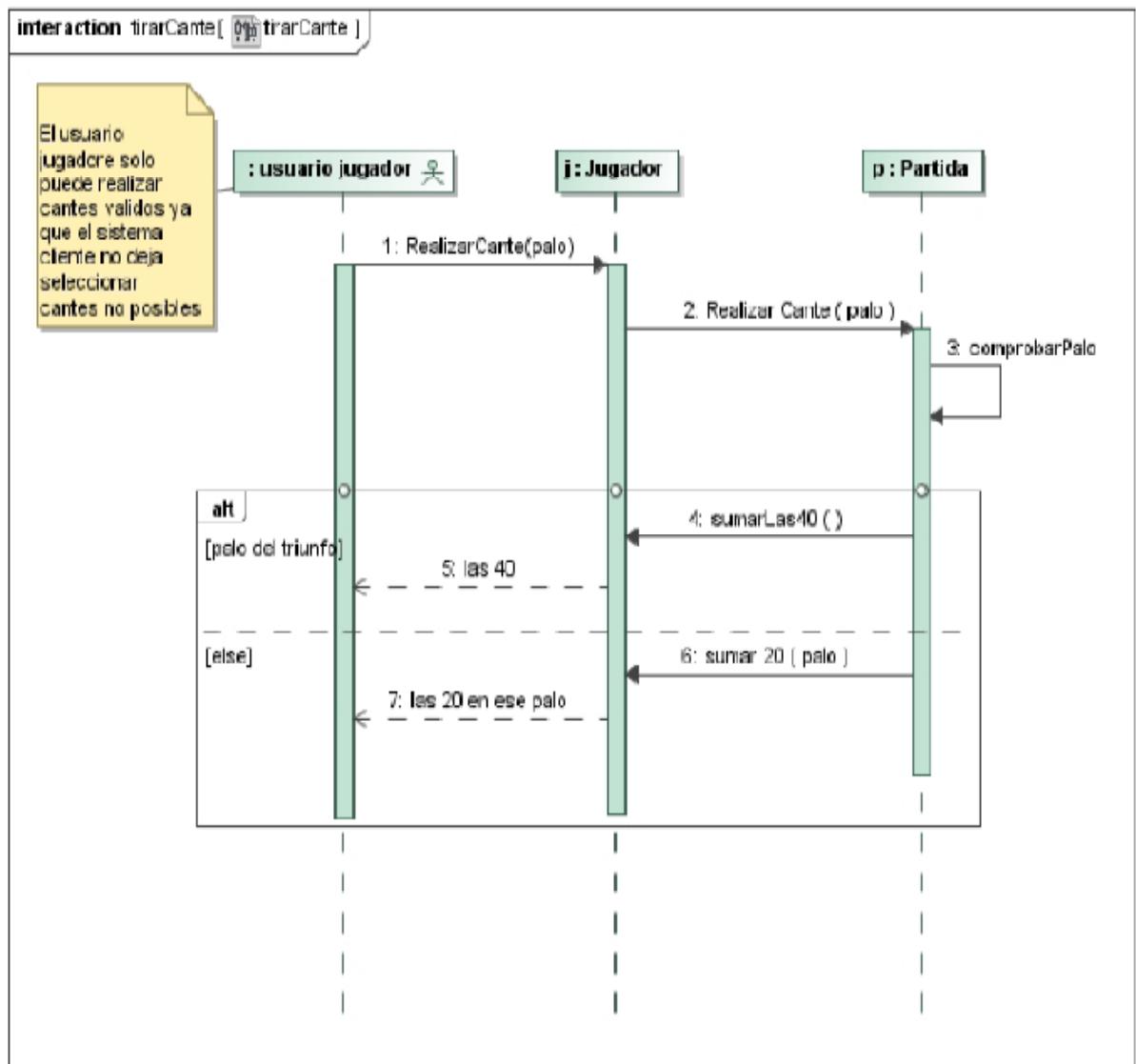
We can observe the sequence of events that occur within the system when a player sits at a table and the card distribution occurs.



9 Diagrama de secuencia: Repartir Cartas

Realizar cante

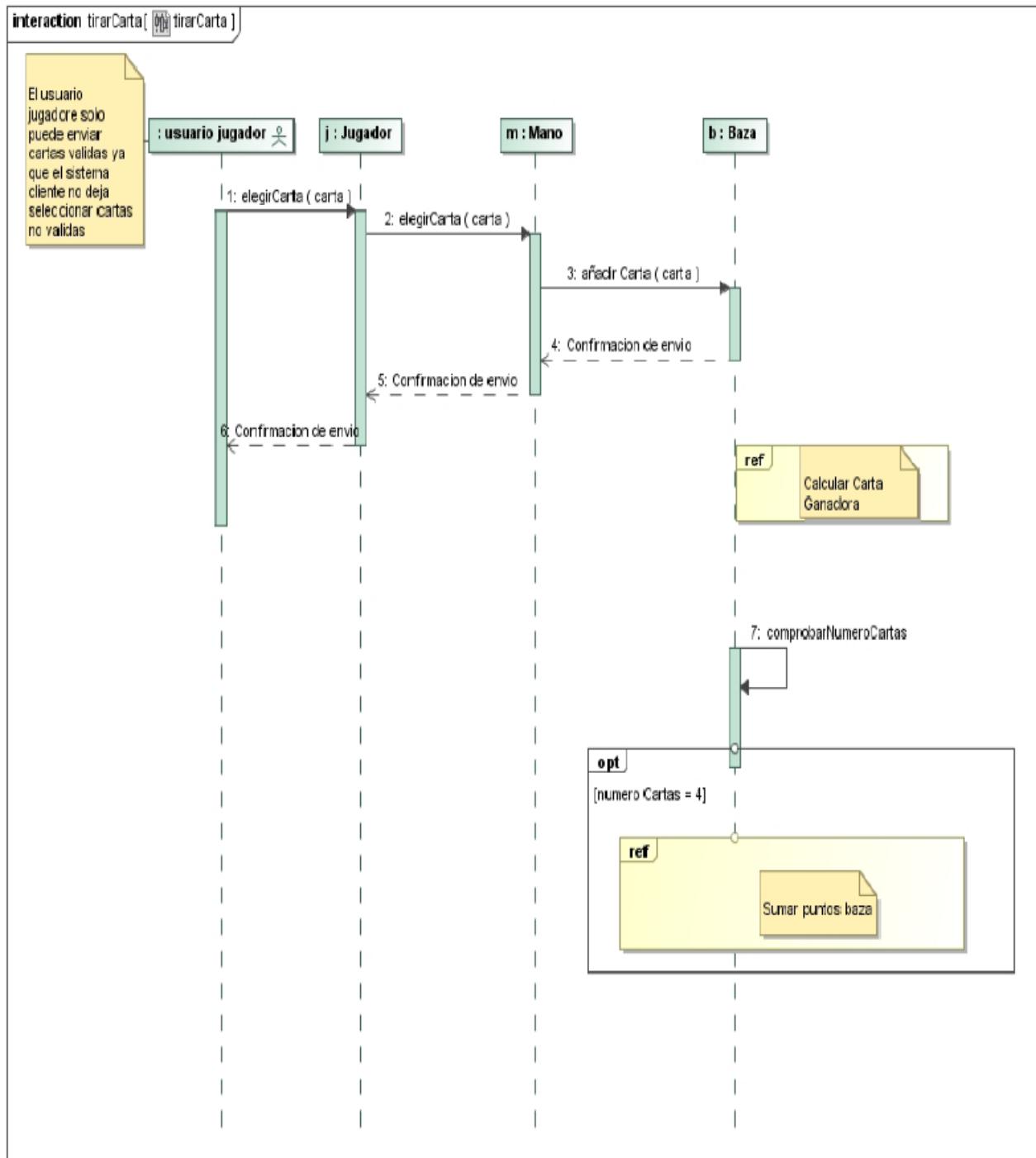
Cuando un usuario realiza un cante, el flujo de eventos que se producen dentro del sistema es el siguiente.



10 Diagrama de secuencia: Realizar cante

Elegir jugada

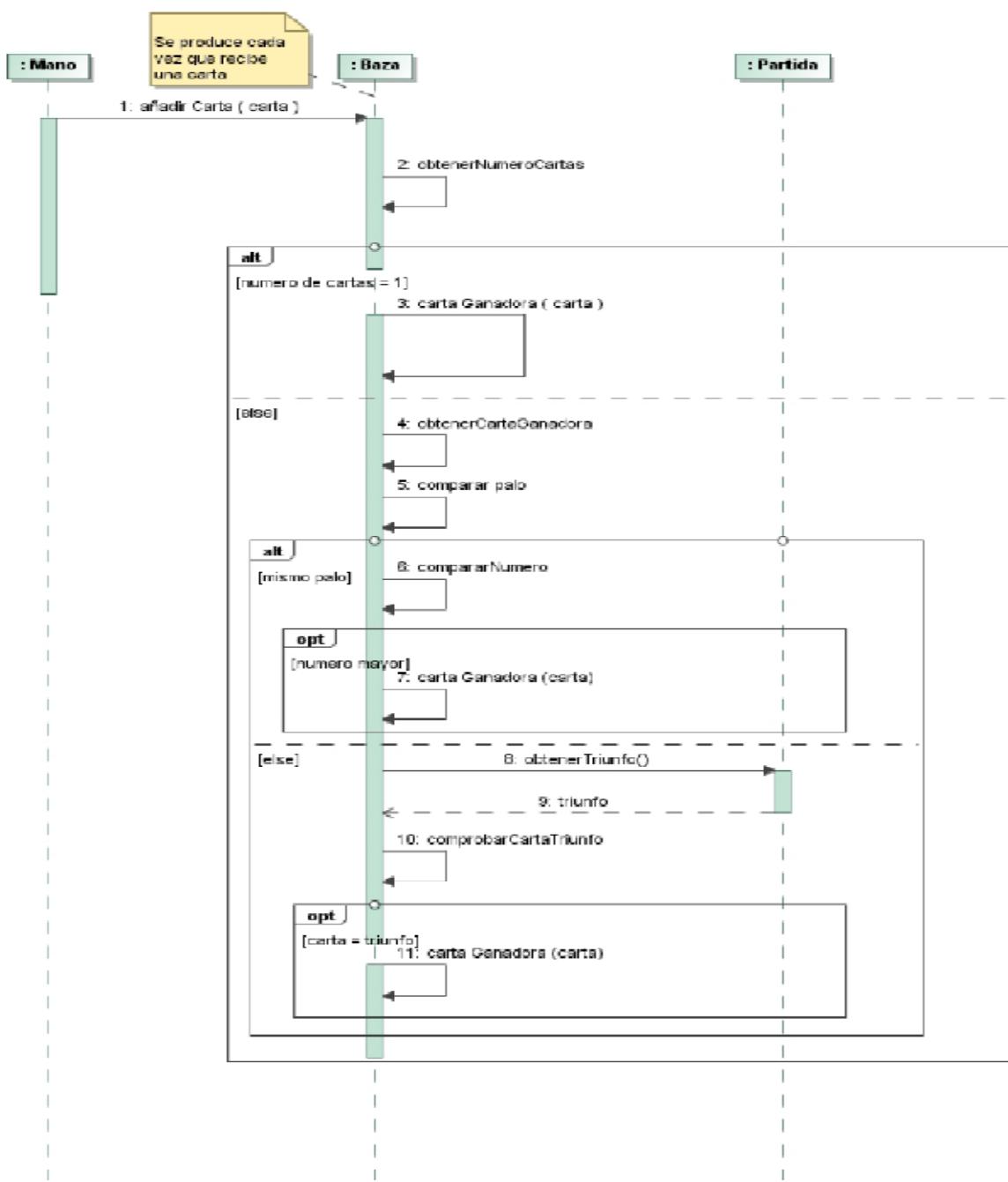
Cuando un usuario realiza la acción de tirar una carta a la mesa, cuando una baza está en juego, el flujo de eventos ocurrido en el sistema es el siguiente.



11 Diagrama de secuencia: Elegir Jugada

Calcular carta ganadora

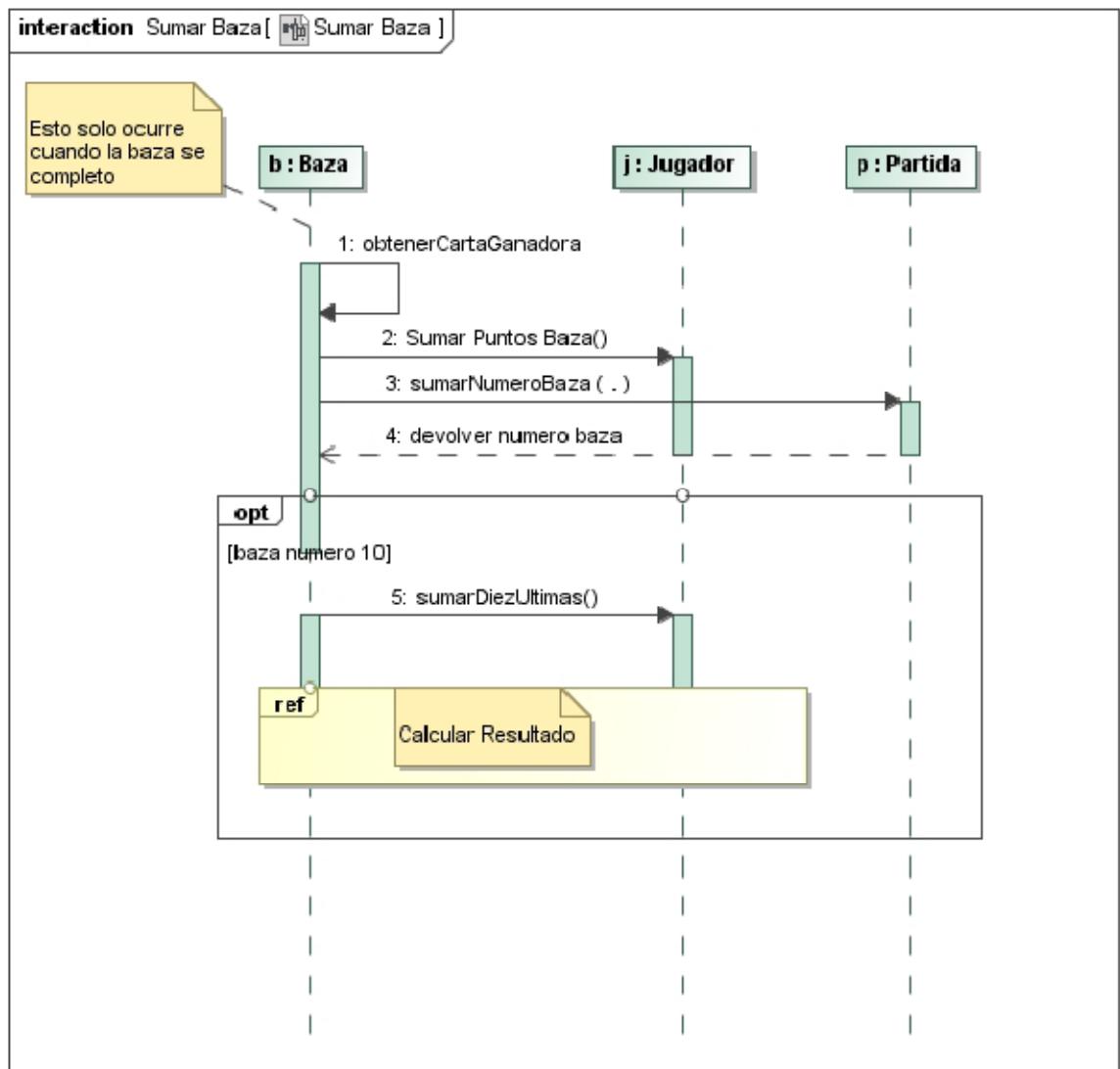
El siguiente diagrama de secuencia va a representar el flujo de eventos que se realiza cada vez que el sistema recibe una carta. Cada vez que reciba una nueva va comparándola con la carta ganadora provisional de la baza, si esta no está vacía, por el contrario, si estuviera vacía marcaría esta ultima como carta ganadora provisional.



12 Diagrama de secuencia: Carta ganadora

Sumar puntos baza

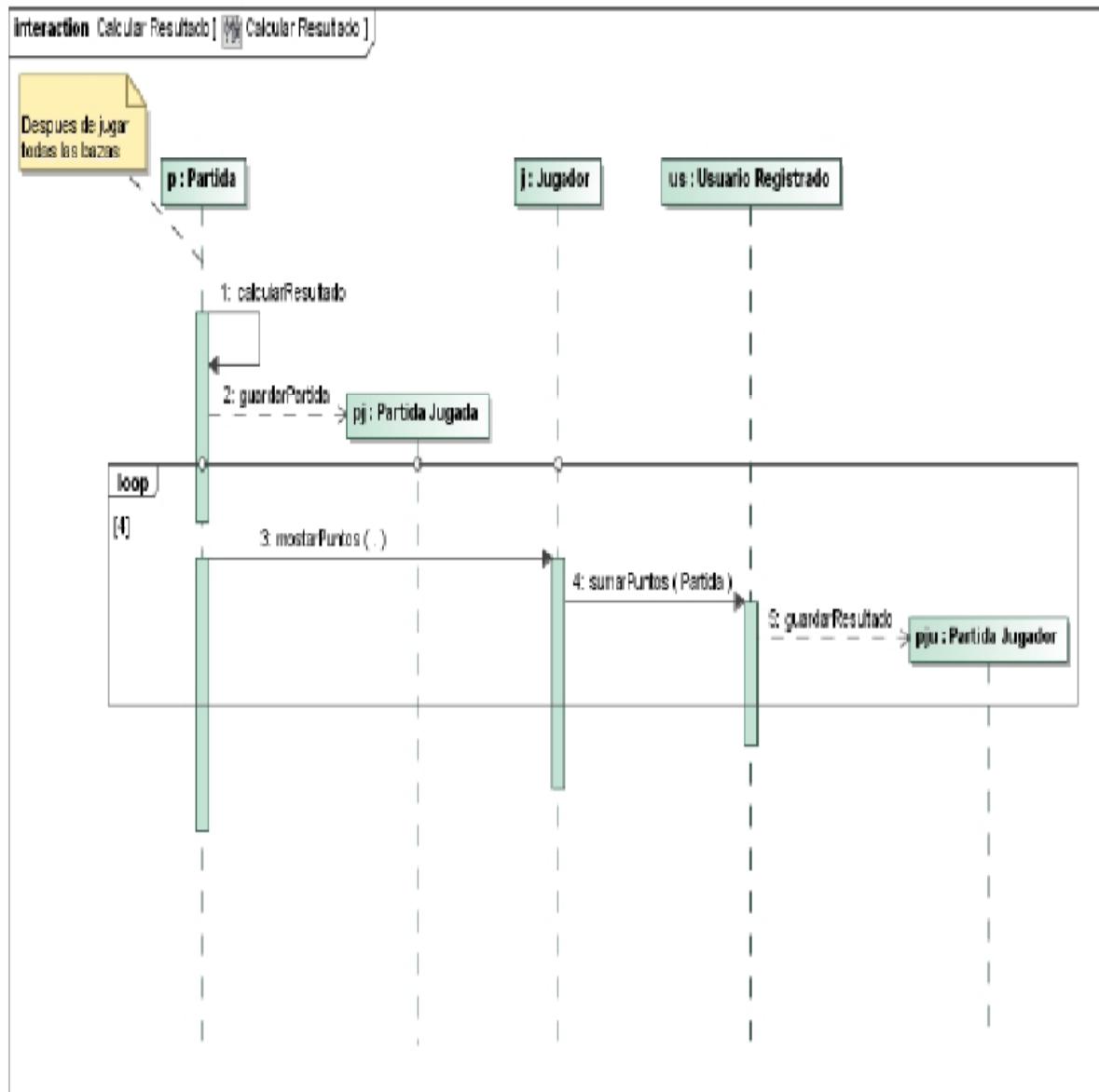
Cuando el sistema recibe una carta y comprueba que es la cuarta carta que contiene la baza, realiza la suma de los puntos a la pareja ganadora de la baza. Esta acción es la representada a continuación.



13 Diagrama de secuencia: Sumar puntos baza

Calcular resultado

Cuando termina una partida, se muestra el resultado a los jugadores, se suma los puntos a los ganadores, y se guarda la partida. El flujo de eventos realizados dentro del sistema es el siguiente.

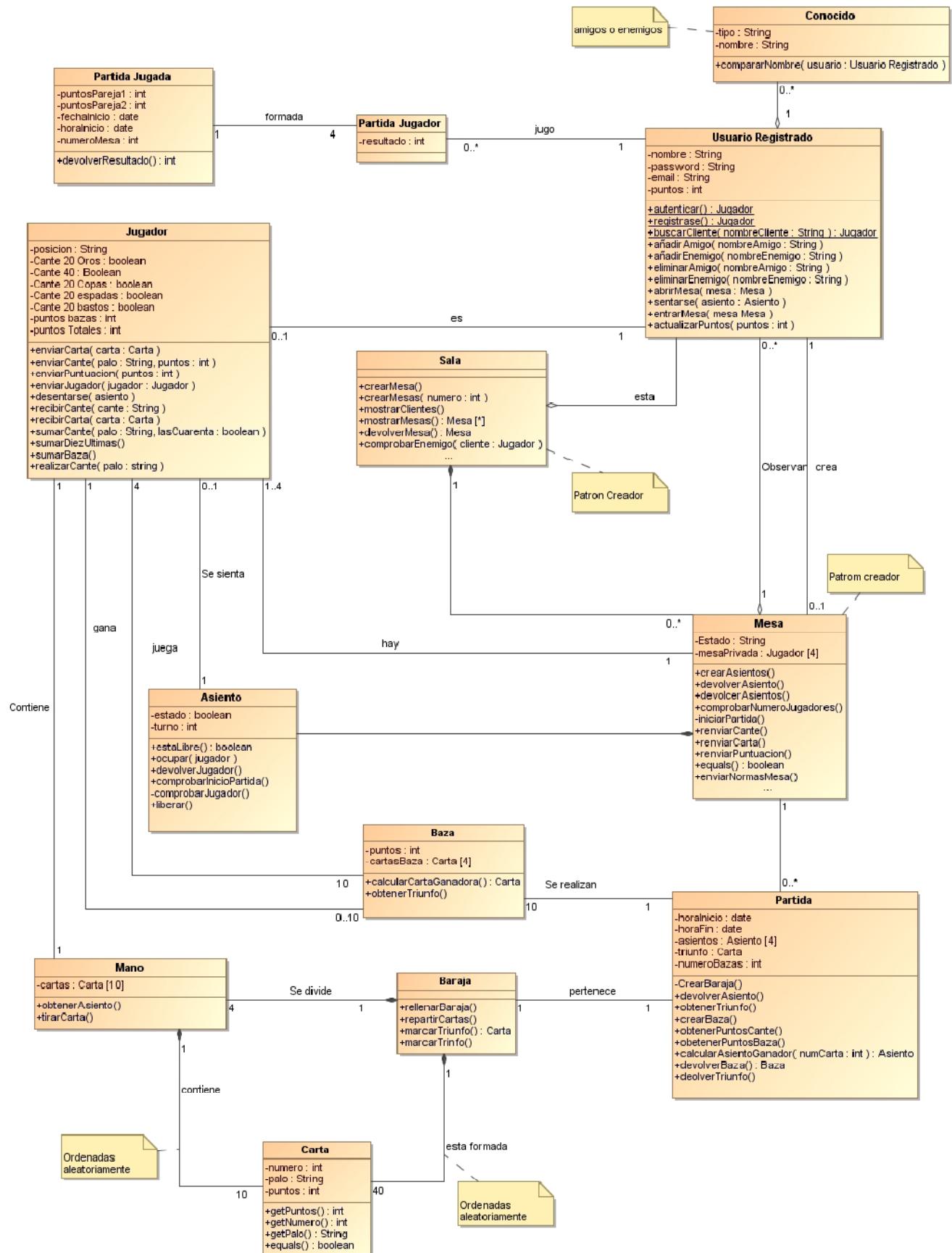


14 Diagrama de secuencia: Calcular resultado

5.2. Modelo clases software

Después de haber realizado el diseño dinámico, continuamos con el diseño estático. Para ello hemos realizado el diagrama de clases software, que representan las diferentes clases que vamos a implementar y en las que se marcan todos sus atributos y sus métodos más importantes.

Desarrollo de una aplicación que permita jugar, en tiempo real, por internet al Tute



15 Diagrama de clases software

5.3. Diagrama de despliegue

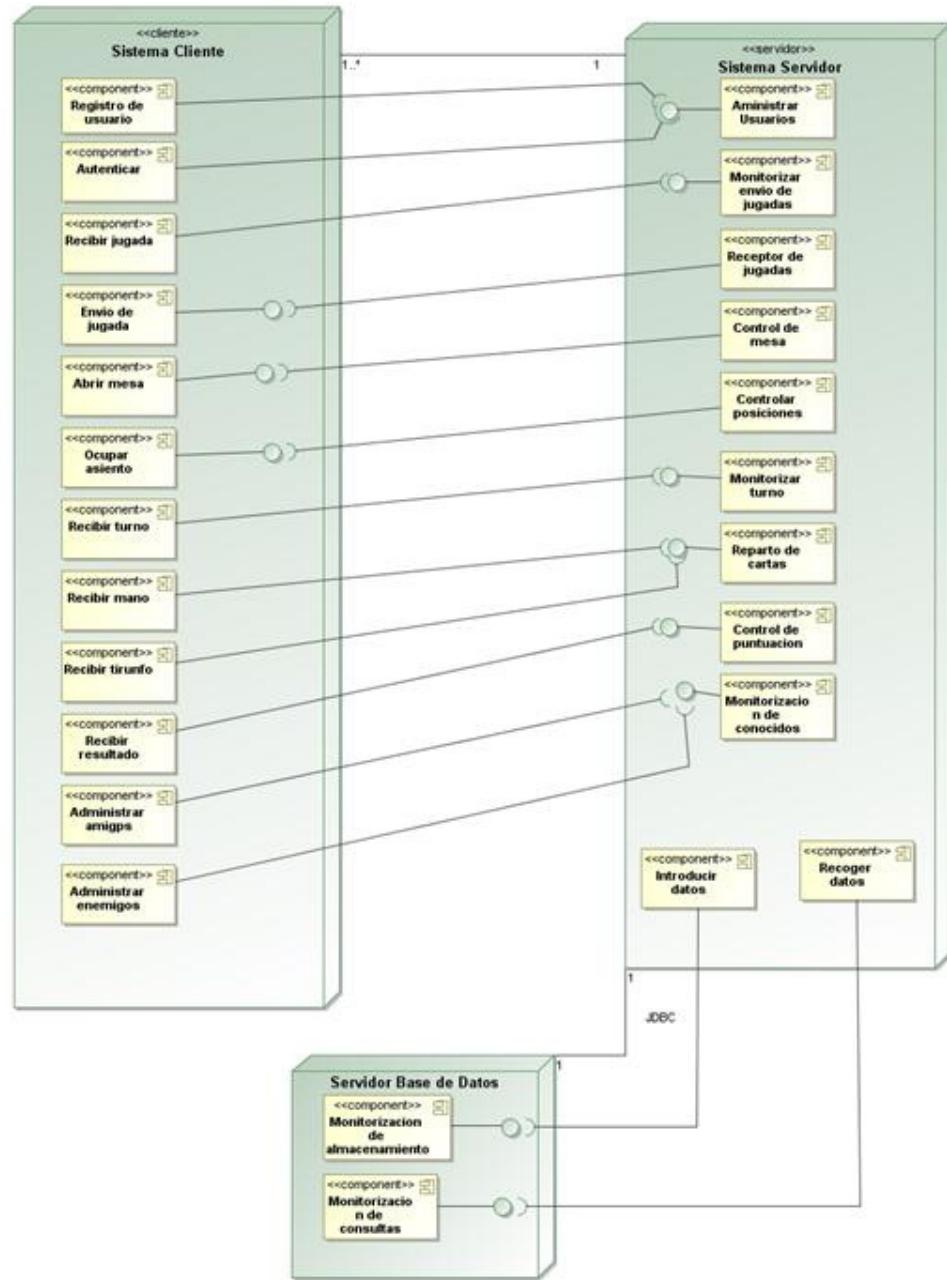
Los diagramas de despliegue se utilizan para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes.

Los usos más comunes para este tipo de diagrama son el de sistemas empotrados, cliente-servidor, completamente distribuidos.

En el caso de los sistemas cliente-servidor, son un extremo del espectro de los sistemas distribuidos y requieren tomar decisiones sobre la conectividad de red de los clientes a los servidores y sobre la distribución física de los componentes software del sistema a través de nodos.

A continuación mostramos el diagrama correspondiente a este proyecto.

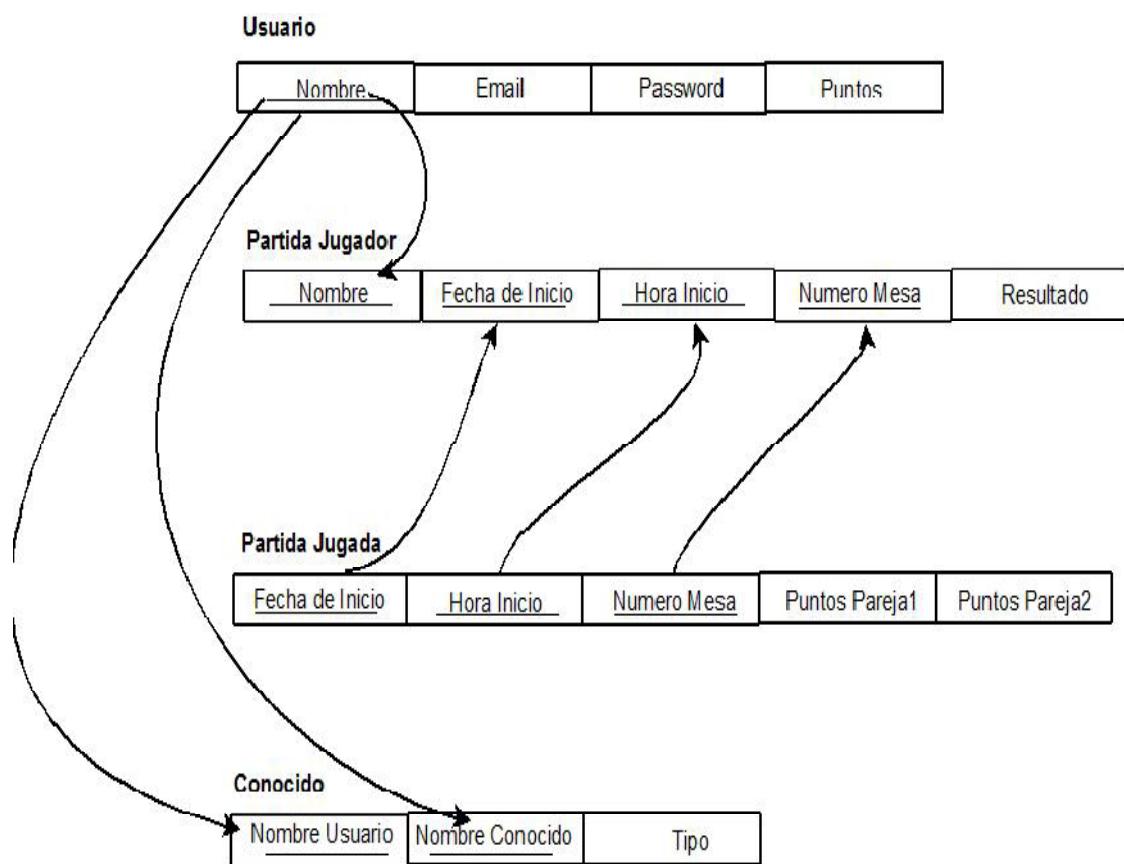
Desarrollo de una aplicación que permita jugar, en tiempo real, por internet al Tute



16Diagrama de despliegue

5.4. Modelo Relacional de la base de datos

A continuación podemos ver el modelo relacional que muestra la estructura y relaciones que tendrán las tablas en la implementación física de la base de datos. El modelo de entidad-relación no fue realizado, ya que un buen diseño del diagrama software nos valdría para usar como entidad-relación. Por supuesto en el modelo relacional solo vamos a mostrar las entidades que deben ser guardadas ya que la mayoría de las clases software no tienen que serlo de forma persistente.



17Modelo relacional

6. Implementación

En este apartado vamos a explicar la implementación en los diferentes incrementos realizados. Esto no representa que solo la implementación fuera creada de forma incremental, sino que cada uno de ellos corresponderá a un incremento en análisis, diseño y implementación.

6.1. Primer incremento

En el primer incremento se realizó la implementación de lo que se consideró el núcleo del proyecto, como es poder realizar una partida de cuatro jugadores. En este incremento, la forma de manejar el sistema era simplemente por líneas de comandos, el sistema le pedía que fuera introduciendo los datos necesarios en cada momento, y la elección de la jugada o cante dentro de una lista de jugadas y cantes válidos, mostrada anteriormente. Si el usuario pulsa un cante no válido, tanto porque no tiene las cartas necesarias o en el momento pulsado no está permitido realizar el cante, se mostrará un mensaje informativo. Pasaría lo mismo en caso que este último pulsara una jugada no permitida en la que está en juego.

Los usuarios simplemente tenían que introducir el nombre y ya estaba directamente sentado en una mesa de juego, esperando por el resto de jugadores. Si intentaba entrar un quinto jugador no se le permitía mostrándole un mensaje de error. También existe el control de que un usuario sea único.

Así como se acaba una partida, las aplicaciones se cierran.

6.2. Segundo incremento

El objetivo principal de este incremento fue realizar la interfaz gráfica de la aplicación desarrollada anteriormente. La interfaz gráfica fue implementada utilizando el patrón MVC (modelo-vista-controlador), su primera imagen nada más ser ejecutado el cliente muestra un título y la opción de insertar el nombre para poder entrar. Una vez el usuario

introduzca el nombre, si no ocurre ningún error nombrados en el incremento anterior, como que ya existen 4 usuarios conectados o el nombre ya está en uso, a continuación se mostrara una ventana que representará la mesa de juego. También se ha implementado en este incremento una nueva funcionalidad, ésta consiste en que al finalizar una partida al usuario se le da la opción de volver a jugar.

6.3. Tercer incremento

En este incremento se ha abordado la opción de que un usuario pueda elegir el asiento donde quiera colocarse, dentro de un número de mesas previamente creadas en el servidor, por lo que también se permitirá el juego de varias partidas simultáneamente. Aparte de estas opciones se contempló todas las diferentes variaciones que estas pueden crear, distintos errores como que un asiento estuviera ocupado.

Un usuario tendrá la opción de salirse de la mesa y volver a la sala en cualquier momento. La consecuencia de esta acción, en caso de que la partida estuviera ya iniciada, será el reinicio, y los demás jugadores serán notificados y quedarán a la espera de que un nuevo usuario entre en la mesa.

En la interfaz gráfica, en la ventana que representa la sala, aparte de mostrarse las mesas disponibles, se mostrará una lista con el nombre de los usuarios conectados en cada instante. Esta interfaz se actualizará automáticamente tanto en el estado de los distintos asientos, como en la lista de usuarios conectados.

De igual manera cuando se termina una partida se dará la opción de volver a ocupar el mismo asiento, o salirse de la partida, todo esto con las diferentes notificaciones al resto de usuarios.

6.4. Cuarto incremento

El cuarto incremento consistió en la implementación del chat de lobby, como también el chat de la mesa de juego. La diferencia entre los dos, es simplemente al público que está dirigido, mientras el chat de lobby puede ser utilizado por todo usuario conectado, el chat de mesa solo puede ser utilizado por los jugadores situados en una mesa. Cada uno está situado dentro de la interfaz gráfica en sus ventanas correspondientes, mientras el chat de lobby está situado en la ventana que representa a la sala, el chat de mesa en está situada en la ventana referente una mesa.

6.5. Quinto incremento

En este incremento, he abordado que un jugador pueda entrar a una mesa como observador, es decir como público. Estará capacitado para poder ver todas las cartas de todos los jugadores, ver las jugadas elegidas, y por otra parte podrá ver lo que se escriben los usuarios por el chat pero ellos no podrán participar en él.

Los usuarios podrán entrar como observadores de una mesa antes de que en ésta de comienzo una partida, es decir, que estén situados cuatro usuarios en ella.

6.6. Sexto incremento

Por otro lado en este apartado se ha realizado la implementación de registro de usuarios mediante la conexión a una base de datos con la tecnología JQDBC. En la aplicación cliente se controlarán todos los posibles errores en la introducción de los datos, como que el nombre deba tener entre 3 y 20 caracteres, la contraseña entre 6 y 80, que coincidan la contraseña y la repetición de la contraseña, y que el email contenga el carácter @.

Igualmente también se implementó la autenticación. Ambas acciones sin utilizar ningún algoritmo de encriptación.

6.7. Séptimo incremento

El séptimo incremento se basa en la realización del caso de uso en el cual un usuario abre una mesa de juego pudiendo reservar los diferentes asientos a distintos jugadores y cubriendo la opción de que una mesa pueda ser visible para el resto de usuarios, restringiendo de este modo su posible participación.

Cuando un usuario crea una mesa, éste se convierte en el anfitrión de la misma. Al abrir una mesa, el usuario tiene la opción de elegir su asiento poniendo su nombre en el asiento correspondiente, en caso de que sea así, cuando se acepta la acción, este usuario ya se sentara automáticamente en

él, por el contrario, si no se sitúa en ningún asiento, se mostrarán la lista de las mesas incluyendo esta nueva.

Los usuarios que fueron invitados, en caso de que estén conectados, se les mostrará la información y la posibilidad de aceptarla o rechazarla, y en el caso de que no estén conectados será la primera opción que se le muestre nada más se autentifiquen.

Se realiza el control de posibles errores a la hora de abrir una mesa, que los nombres de los usuarios introducidos sean válidos y que no se repita el mismo nombre en distintos asientos.

Por último, en este incremento se realizó la posibilidad de cambiar de anfitrión en caso de que el antiguo se desconecte. Esta elección la realiza el servidor marcando como anfitrión a uno de los usuarios que se encuentren en la mesa, en caso de que la mesa se encuentre vacía, ésta será eliminada.

6.8. Octavo incremento

Se trata de un incremento en el cual fue mejorado algún funcionamiento y aplicado un algoritmo de encriptación para la contraseña utilizada por los usuarios.

Con las mejoras nos referimos a que ahora un usuario podrá entrar como público en una mesa en cualquier momento, esté la partida empezada o no.

El algoritmo empleado en la encriptación de la contraseña es el conocido algoritmo SHA-1, que será explicado en su correspondiente anexo.

6.9. Noveno incremento

En este último incremento se ha realizado la posibilidad de consultar el ranking de puntuaciones de los usuarios; éstos recibirán 10 puntos por cada partida ganada y se restará 5 puntos por cada partida perdida.

En el ranking se mostrará la posición, el nombre del usuario, los puntos, el número de partidas ganadas y el número de partidas perdidas.

Para esto, se han automatizado las inserciones de los datos de las diferentes partidas, y se ha realizado la correspondiente consulta para poder realizarlo.

7. Pruebas

Para comprobar que las distintas funcionalidades de la aplicación se realizan correctamente se han llevado a cabo una serie de pruebas. Se han hecho las correspondientes pruebas de unidad y por último las pruebas de integración.

7.1. Pruebas de unidad

Se han realizado pruebas en cada clase “java” para comprobar que cada una funcionaba bien por sí sola, sin interacción con las demás clases del sistema.

7.2. Pruebas de integridad

Para la realización de las pruebas de integración no se ha utilizado ninguna herramienta que automatice la tarea. Para realizarlas, se ha ejecutado y comprobado el funcionamiento en diferentes ordenadores con diferentes resoluciones y diferentes sistemas operativos. Estas pruebas se han realizado a través de internet donde los diferentes usuarios en sus respectivas máquinas han ejecutado la aplicación cliente y todas se han conectado a la misma aplicación servidor. Estos usuarios realizaron todas las funciones posibles que ofrece la aplicación, y se han comprobado las diferentes respuestas del servidor.

8. Manual de Usuario

El manual de usuario sólo vamos a aplicarlo a la aplicación cliente, ya que en la parte servidor está todo automatizado y no puede ser manejado por una persona para nada más que iniciarla.

8.1. Autenticarse

Nada más iniciar la aplicación saldrá la ventana de inicio mostrada a continuación:



18 Ventana: Autenticarse

Como se puede ver está compuesta por una barra de estados (explicada más adelante), el título de la aplicación y a continuación la zona para insertar los datos necesarios para autenticarse. Como se puede observar encima del botón para entrar, se puede ver un mensaje que pone “Si está registrado” ya que una persona que no se haya registrado no puede autenticarse.

Más abajo en rojo, para ser más visible, donde se puede leer “Si no está registrado pulse aquí” es donde debe pulsar un usuario no registrado para realizar su registro.

En la siguiente imagen se puede ver un error dado a la hora de autenticarse.

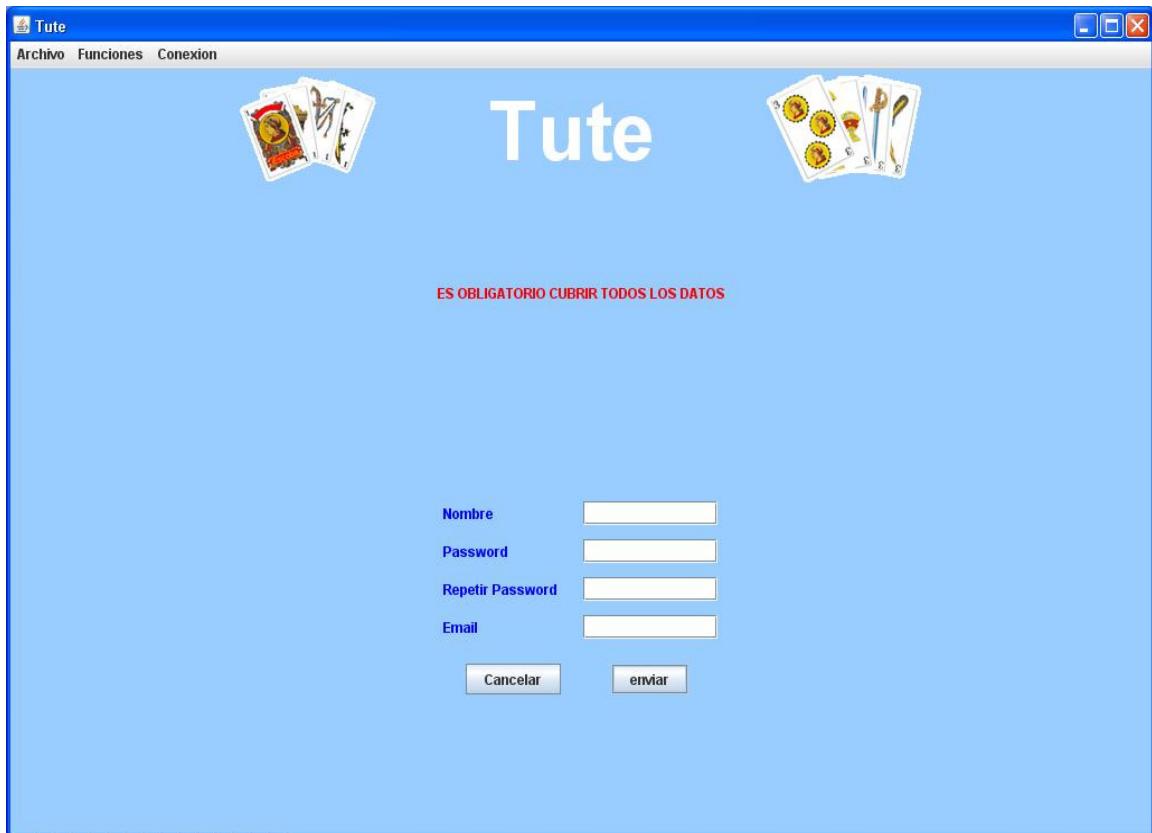


19Ventana: Autenticarse "Error"

El error es marcado encima de la zona de introducir los datos para autenticarse, en este caso el error es “El usuario o password es incorrecto” y es donde aparecerán los errores posibles a la hora de autenticarse, que pueden ser este y “El usuario ya está conectado”.

8.2. Registrarse

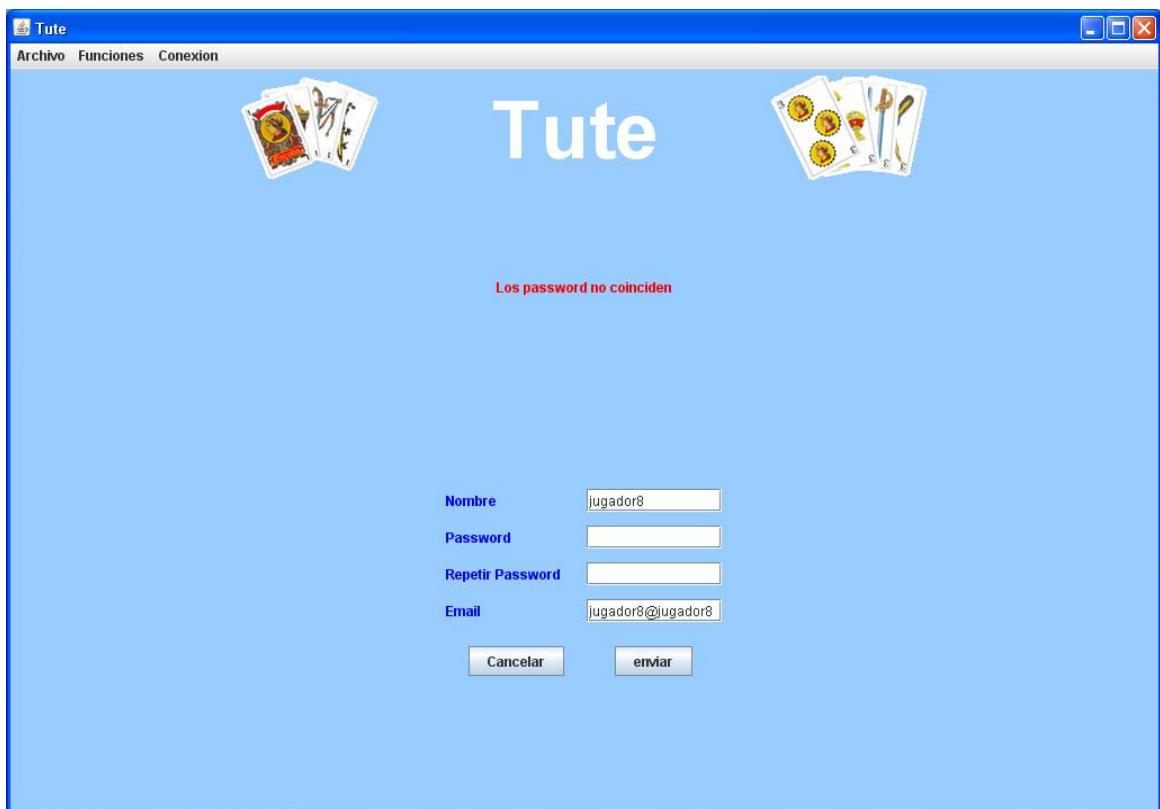
Esta es la imagen de la apariencia que tendrá la ventana cuando un usuario pulsa la entrada anteriormente mencionada que ponía “Si no está registrado pulsa aquí”:



20Ventana: Registrarse

En la imagen se puede ver en primera parte la barra y el título ya conocido, a continuación un mensaje en rojo donde pone “Es obligatorio cubrir todos los datos”, que este mensaje cambiará en caso que se produzca un error, y por último la zona para cubrir los datos necesarios y los botones correspondientes para cancelar y enviar los datos.

A continuación se muestra una imagen donde se puede leer un error producido:



21Ventana: Registrarse "Error"

En este caso ocurrido es “Los password no coinciden” y se puede ver que los datos anteriormente introducidos siguen marcados, menos los incorrectos.

Cuando un usuario introduce los datos correctamente y el nombre elegido no está escogido por otro usuario saldrá el panel de inicio mostrando un mensaje de registro realizado correctamente y el usuario ya podría iniciar sesión automáticamente.

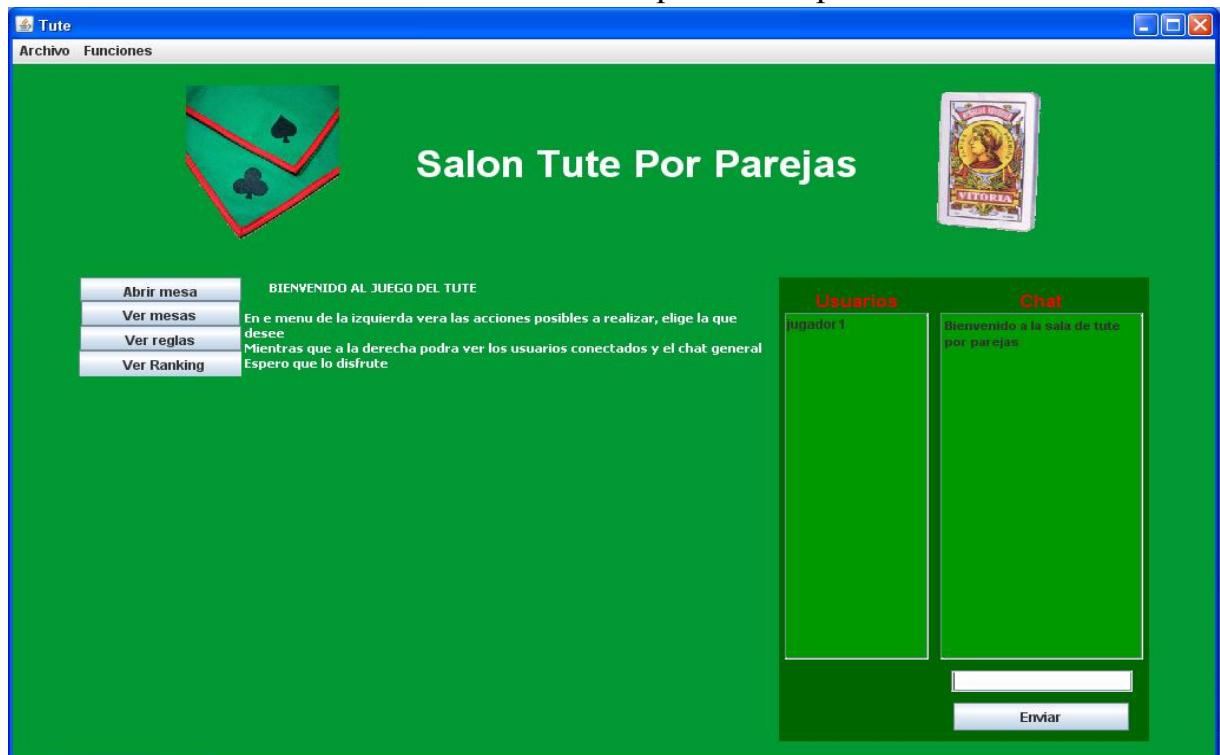
Desarrollo de una aplicación que permita jugar, en tiempo real, por internet al Tute



22 Ventana: Registrarse "Confirmación"

8.3. Iniciada la sesión

Una vez autenticado la ventana adquiere esta presencia:

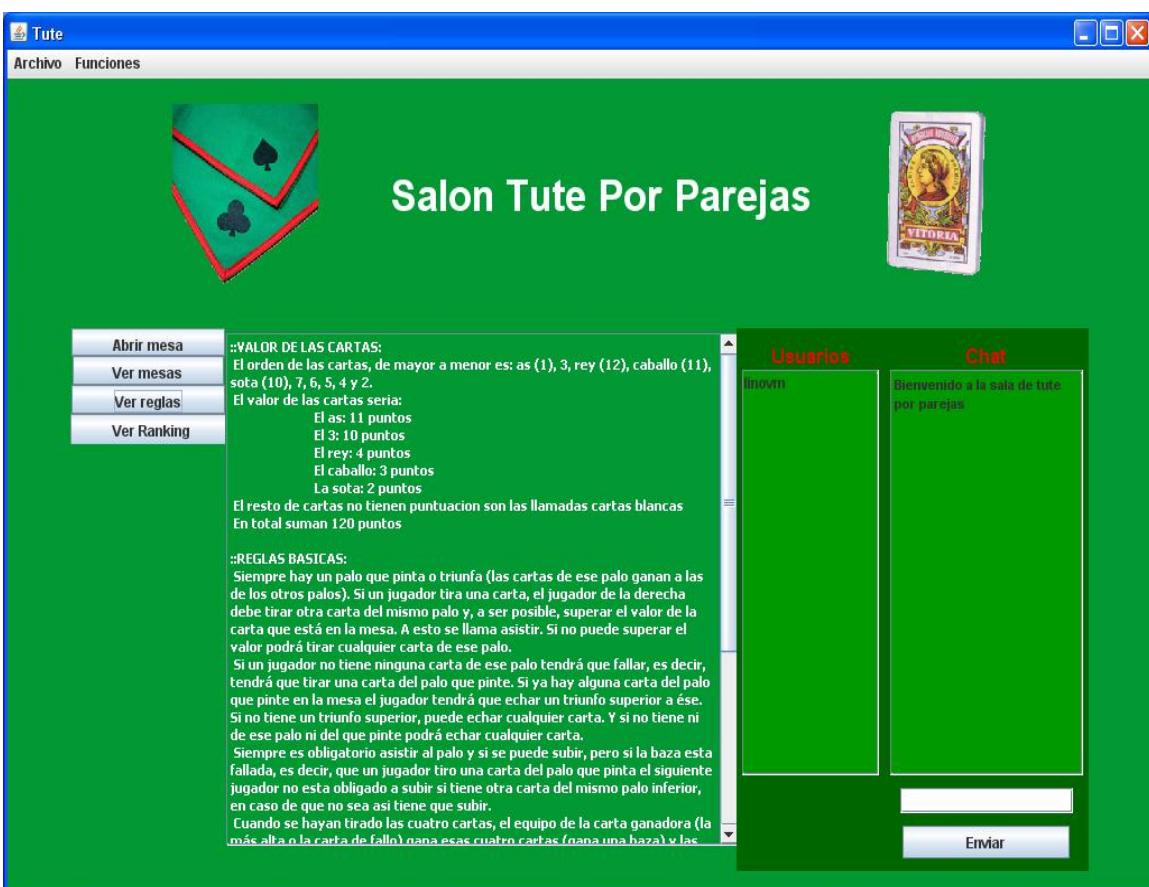


23 Ventana: Bienvenida

En primer lugar la barra de tareas comentada anteriormente y explicada más tarde, el título de la sala, y después el resto se divide en tres columnas. La primera columna son los botones correspondientes a las distintas operaciones que puede realizar el usuario: Abrir mesa, ver mesas disponibles, ver reglas y ver el ranking. La columna del medio que será la que varíe según la operación elegida por el usuario mostrará un mensaje de bienvenida. A la derecha se encontrará una lista de los usuarios conectados y el chat de lobby.

8.4. Ver reglas

Cuando se presiona el botón de ver reglas se mostrarán las diferentes reglas del juego del tute por parejas de la siguiente forma.



24Ventana: Reglas

Las reglas mostradas son:

- Valor de las cartas:

El orden de las cartas, de mayor a menor es: as (1), 3, rey (12), caballo (11), sota (10), 7, 6, 5, 4 y 2.

- El valor de las cartas seria:

El as: 11 puntos.

El 3: 10 puntos.

El rey: 4 puntos.

El caballo: 3 puntos.

La sota: 2 puntos.

El resto de cartas no tienen puntuación son las llamadas cartas blancas.

En total suman 120 puntos .

- Reglas básicas:

Siempre hay un palo que pinta o triunfa (las cartas de ese palo ganan a las de los otros palos). Si un jugador tira una carta, el jugador de la derecha debe tirar otra carta del mismo palo y, a ser posible, superar el valor de la carta que está en la mesa. A esto se llama asistir. Si no puede superar el valor podrá tirar cualquier carta de ese palo.

Si un jugador no tiene ninguna carta de ese palo tendrá que fallar, es decir, tendrá que tirar una carta del palo que pinte. Si ya hay alguna carta del palo que pinte en la mesa el jugador tendrá que echar un triunfo superior a ése. Si no tiene un triunfo superior, puede echar cualquier carta. Y si no tiene ni de ese palo ni del que pinte podrá echar cualquier carta.

Aunque asistir siempre es obligatorio, hay una excepción en la cual no se está obligado a subir. Esta excepción es cuando antes de llegar tu turno otro jugador realiza un fallo, tira una carta que pinta, entonces no se está obligado a subir en caso de que se tenga una carta del mismo palo aunque sea de un valor más bajo.

Cuando se hayan tirado las cuatro cartas, el equipo de la carta ganadora (la más alta o la carta de fallo) gana esas cuatro cartas (gana una baza) y las guarda para contar su puntuación cuando se hayan jugado todas las cartas.

- Los cantes:

Para cantar hace falta:

El rey y caballo del palo que triunfa o pinta: se cantan las 40.

El rey y caballo de otro palo: se cantan las 20.

Si algún equipo canta las 40 ó 20, se suma esa puntuación a la puntuación total de las cartas.

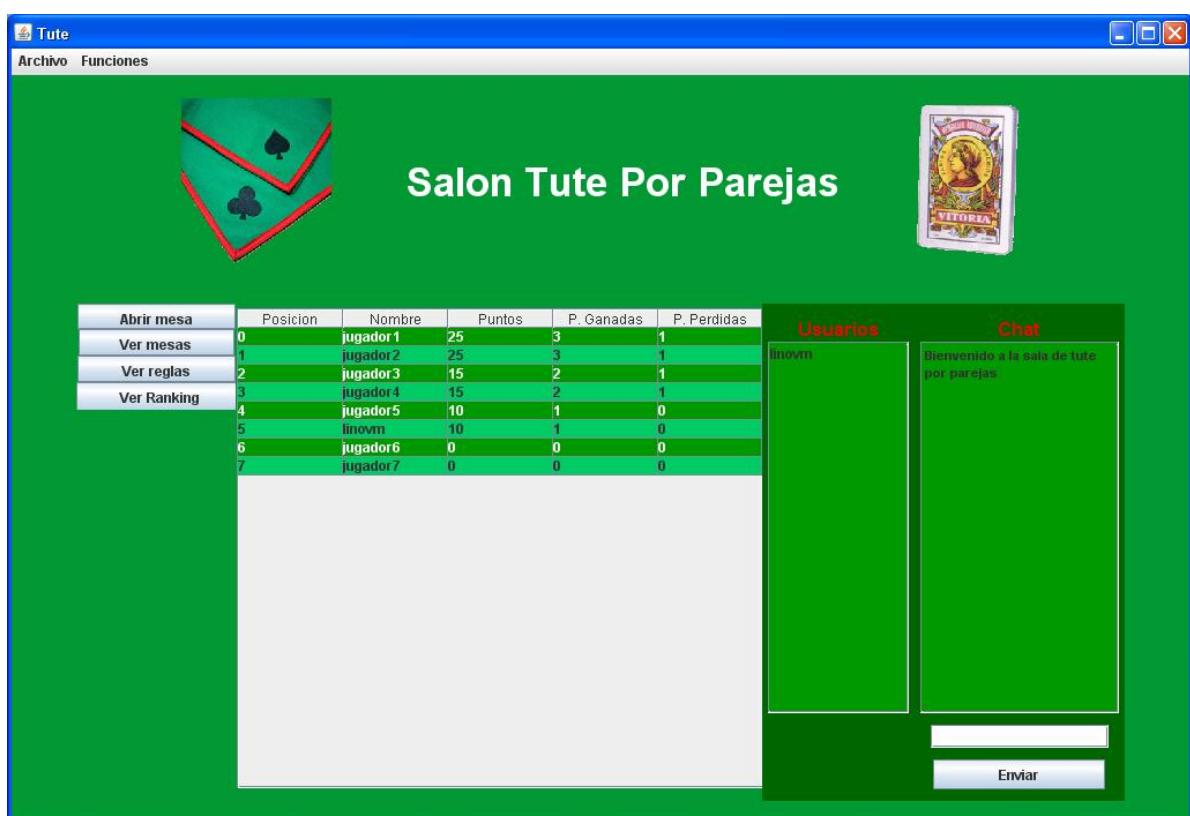
Sólo se puede cantar cuando el equipo al que uno pertenece ha hecho la primera baza. Si se tienen dos o más cantes, se canta la primera vez cuando se consigue la primera baza, otra vez cuando se consigue la segunda, etc. Dando siempre prioridad a las 40.

- Última baza:

Al equipo ganador de la última baza se le suman 10 puntos, aparte si el resultado fuera de empate, a esta pareja se le considerará la ganadora.

8.5. Ver ranking

Si se activa el botón de ver ranking la ventana cogerá la siguiente apariencia:



Se puede ver que cada fila corresponde a un usuario registrado donde se pone la posición que ocupa, el nombre, los puntos obtenidos, el número de partidas ganadas y el número de partidas perdidas.

8.6. Ver mesas

Esta función se refiere a cuando un usuario quiere ver las mesas disponibles o abiertas en ese momento. Las mesas se representarán de esta forma:



26 Ventana: Ver mesas

Para ver mejor la información que se ofrece de las mesas, a continuación se muestra más ampliada la tabla donde se sitúa la información.

Nº Mesa	Anfitrion	A. Norte	A. Sur	A. Oeste	A. Este	Publico
0	Servidor	Libre	Libre	Libre	Libre	Publico
1	Servidor	Libre	Libre	Libre	Libre	Publico
2	Servidor	Libre	Libre	Libre	Libre	Publico
3	Servidor	Libre	Libre	Libre	Libre	Publico
4	Servidor	Libre	Libre	Libre	Libre	Publico
5	Servidor	Libre	Libre	jugador6	Libre	Publico
6	jugador2	jugador2	Libre	Libre	Libre	Publico
7	jugador3	Reservado	Reservado	Libre	Libre	Publico
8	linovm	Libre	Libre	Libre	Libre	Privada

27 Tabla de mesas

Los datos mostrados corresponden a: el número de mesa, el anfitrión de la mesa, el estado de los cuatro asientos y si es posible o no acceder como público.

Cada mesa está representada por una fila y las diferentes celdas pueden tener los siguientes valores:

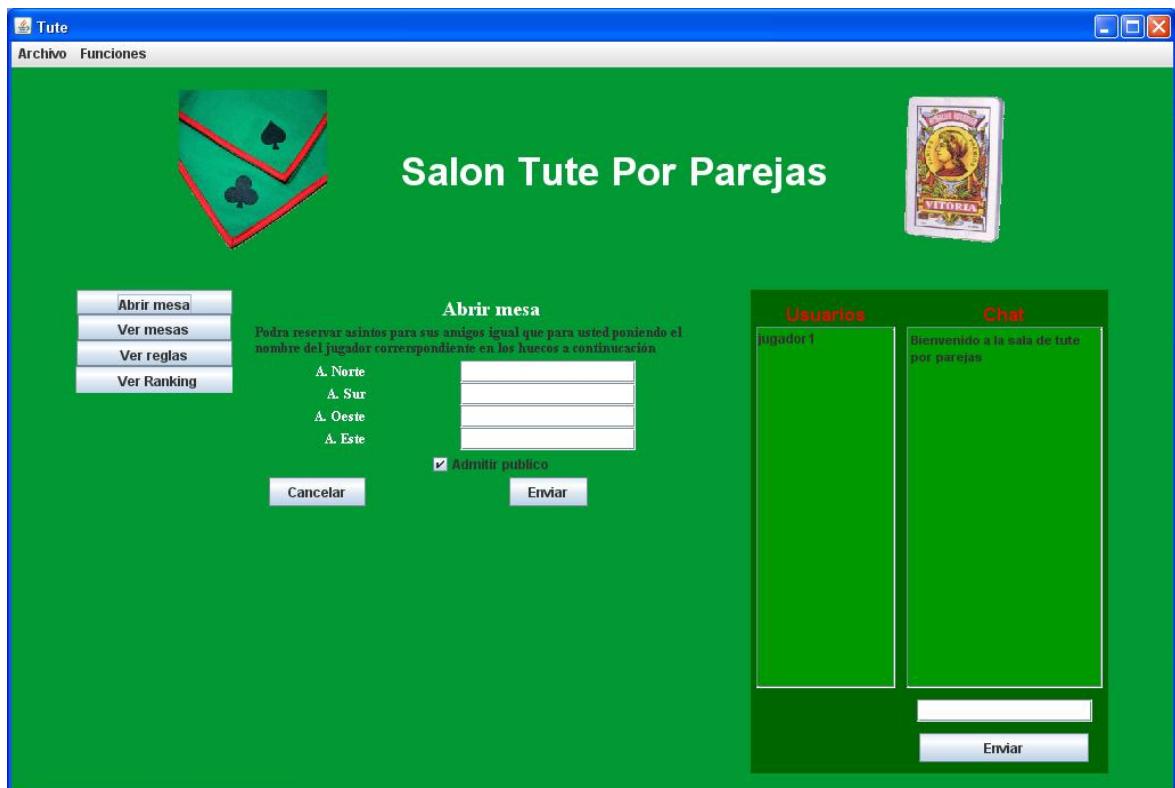
El anfitrión de la mesa muestra quien fue su creador, existen dos variantes, que pueden ser el nombre de un usuario o la palabra “servidor” que significa que es una mesa creada automáticamente al iniciarse el servidor.

La información de los asientos están situadas bajo las columna “A. Norte”, “A. Sur”, “A. Oeste” y “A. Este”. Las parejas estarán formadas, la primera, por los usuarios que ocupen los asientos situados en “A. Norte” y “A. Sur”, mientras la segunda pareja estará formada por los usuarios que ocupen los asiento situados en “A. Oeste” y “A. Este”. El estado de un asiento puede ser de tres tipos: “libre”, que puedes situarte en él simplemente pulsando sobre él y ya entrarás en la mesa, “reservado” que significa que el asiento está reservado para otro usuario, y por último el otro estado posible es el nombre de un usuario que marca quien está sentado en él.

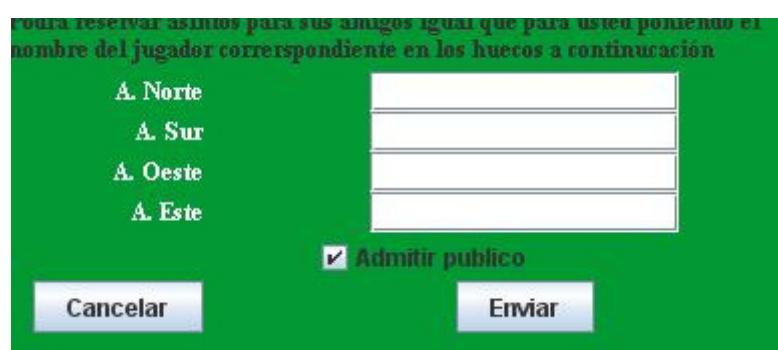
Por último, la columna “público” marca si en la mesa está permitido entrar como observador, señalándola con “publico”, o si no se permite, marcado con “privado”.

8.7. Abrir mesa

Cuando un usuario quiere abrir una mesa la ventana coge el siguiente aspecto.



28 Ventana: Abrir mesa



29 Menu abrir mesa

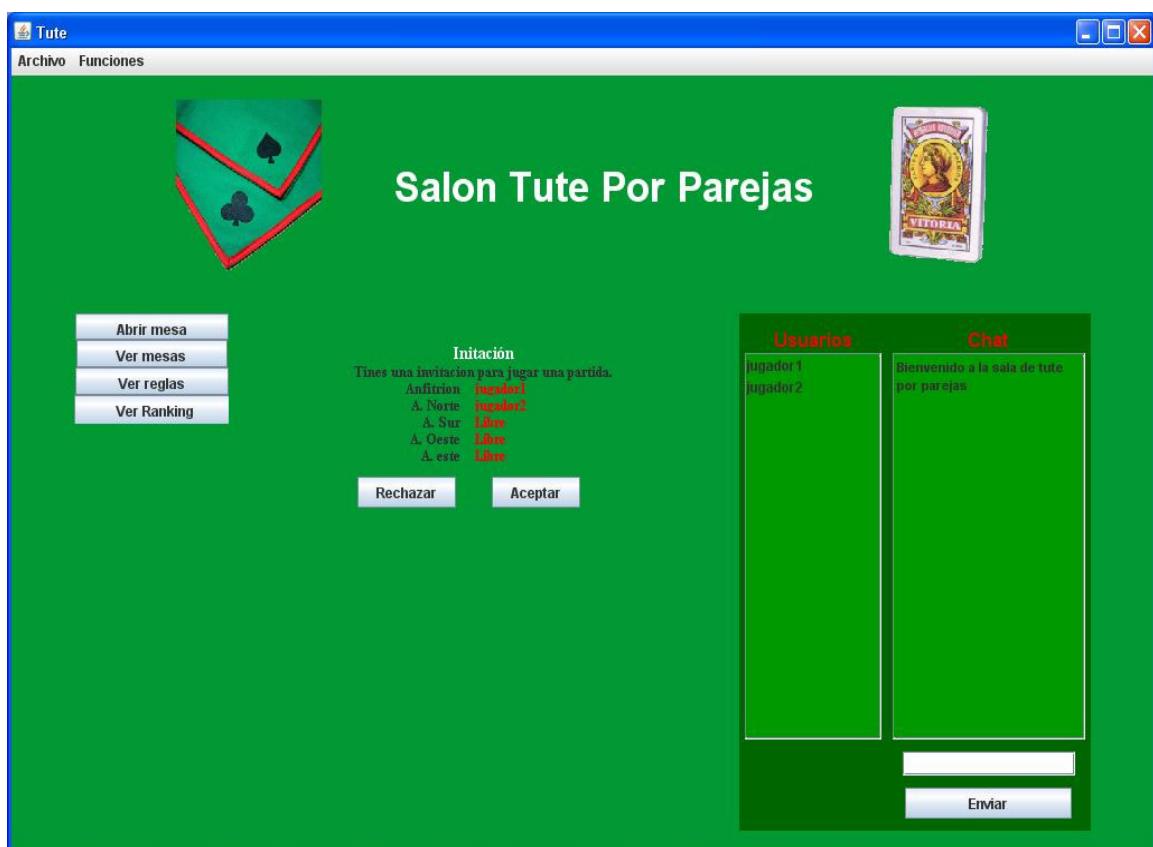
En la imagen se puede ver, en primer lugar, un texto informativo en el cual se comenta que un usuario puede introducir el nombre de un usuario

en el asiento que quiera reservar para ese o introducir el suyo mismo para sentarse él.

Además tiene una casilla para seleccionar si se admite público o no.

Si al abrir la mesa sitúo su nombre en algún asiento, como ya he dicho en el incremento correspondiente, ya nos situaríamos en la mesa, mientras que si no fuera así nos situamos en ver las mesas disponibles, en la cual se vería la nueva mesa creada.

Cuando un usuario realiza una reserva , al usuario correspondiente le aparece un mensaje de invitación, al momento si el usuario está conectado, o será el primer mensaje cuando se conecte. El mensaje que le aparecerá será este:



30 Ventana: Invitacion



31 Menú invitación

Al usuario se le muestra la información de la mesa, es decir, quién ocupará los asientos y cuales están libres, y además el nombre del anfitrión. El usuario tendrá la opción de aceptarla o rechazarla.

8.8. Sentado en mesa

A continuación se va a explicar el panel que representa a una mesa de juego. Primero vamos a mostrar cómo sería cuando uno o varios usuarios están esperando a que la mesa se complete.



En la anterior imagen podemos ver que al lado del nombre de los usuarios pone el asiento que están ocupando.

32 Ventana: Mesa

A continuación vemos cuando la mesa esta completa, las cartas se reparten automáticamente al entrar el último jugador.



33Ventana: Partida

de la mesa, y a continuación vamos a describir dos señales que se pueden ver en la imagen:

- 🔴 El primero es este punto rojo que marca de quien es el turno.
- ⭐ Y de segundo esta estrella que marca quien fue el repartidor.

El resto de características serán explicadas a partir de la siguiente imagen después de haber jugado dos bazas.



34 Ventana: Partida en juego

A continuación se explicara la siguiente imagen por partes.

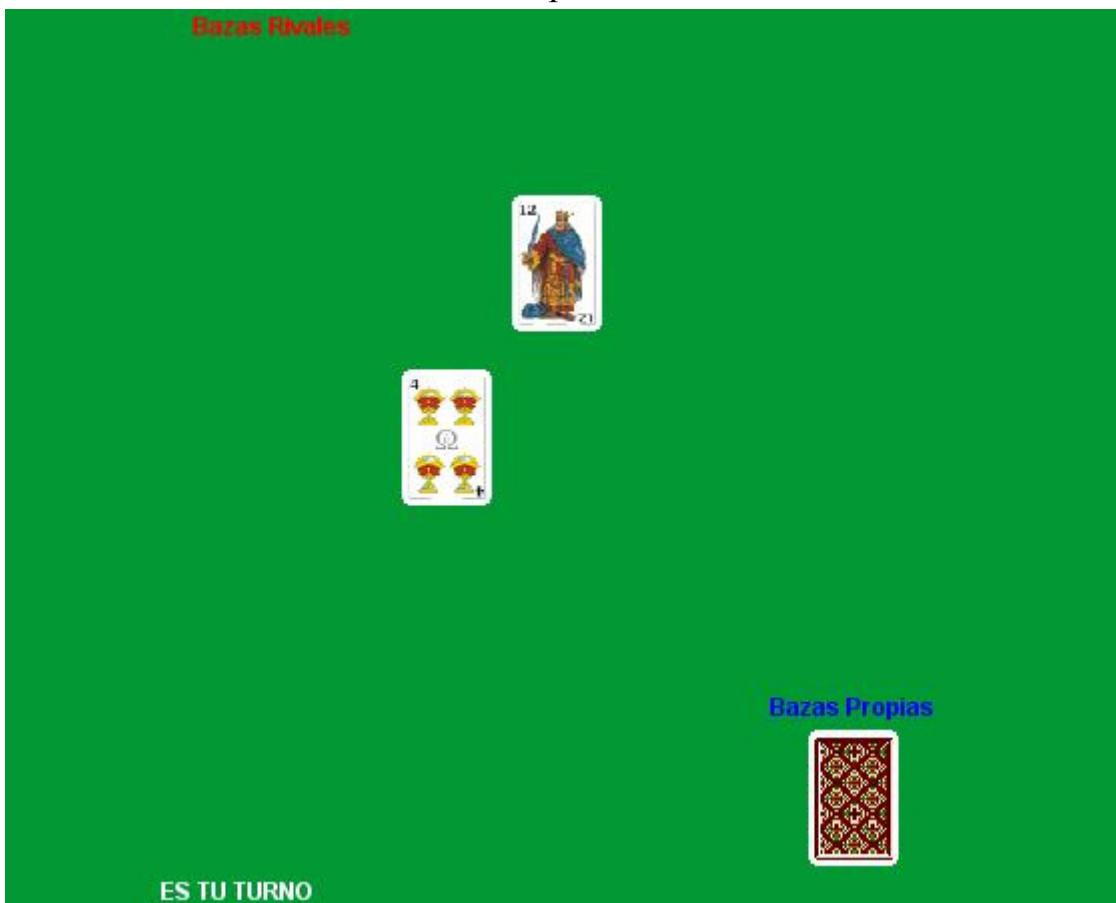
Primero vamos a explicar la sección propia del usuario que está jugando a través de esta ventana:



35Menu jugador

Podemos observar que el usuario dispone de un botón el cual le quitará de la partida y saldrá al salón. A continuación tendrá sus cartas en las cuales podrá elegir la jugada que prefiera, si la jugada no fuera válida se le mostraría un mensaje de error (en la siguiente imagen mostraremos el lugar). Y por último podemos observar los cantos que existen en la partida y que el usuario deberá marcar para realizarlos, si el cante no es válido por los diferentes motivos posibles, igualmente que en elegir la jugada, se mostrará un mensaje.

A continuación se describirá la parte central de la ventana:



36 Centro de una mesa

En este caso podemos ver que las bazas se las llevó la misma pareja ya que vemos las bazas rivales situadas en la esquina de la superior izquierda está vacía mientras que la contraria tiene la baza marcada. También podemos observar el lugar donde se situarán los distintos mensajes, que en este caso el mensaje que se puede leer “Es tu turno”. Por último vemos donde se va a ir situando las cartas jugadas en las bazas.

La última parte a definir es el menú de la derecha donde podremos ver las diferentes estadísticas de la partida como son hora de inicio, duración,

repartidor, triunfo, última baza jugada, ganador de la última baza, puntuación y el chat de mesa.



37Menu estadistica de una partida

8.9. Fin de la partida

A continuación se va a mostrar la ventana cuando una partida finaliza. En ella se podrá ver la hora de inicio, la duración y la puntuación de las dos parejas. Después se le dará la oportunidad al jugador de elegir entre volver a la sala o volver a jugar.



38 Ventana: Fin partida

8.10. Entrar como observador

La última imagen a mostrar y explicar de la aplicación es la vista que tendría un usuario que entre en una mesa como observador.

Se puede ver que el usuario está capacitado para ver todas las cartas de los usuarios, el repartidor y el turno estarán marcados por la estrella y el

punto rojo. Los mensajes se muestran en la misma situación que si se entrara como jugador, y el menú derecho muestra lo mismo que en el anterior menú.

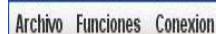
La única diferencia es que un observador no podrá escribir en el chat, y el botón para salir de la mesa está situado al final del menú derecho.



39Ventana: Observador

8.11. Barra de tareas

A continuación se explicaran todos los menús que se encuentran en la base de datos y las funciones que se pueden realizar desde el.



40 Barra de tareas

Comprobamos que existen tres menús, “archivo”, “funciones” y “conexión”. En el primer menú, Archivo, se podrán encontrar dos funciones: desconectar y salir. La primera función, desconectar, tendrá efecto solo cuando un usuario se haya autenticado, y producirá el cierre de su sesión volviendo a la ventana de inicio donde se podrá autenticar con la misma cuenta o otra. La segunda función, salir, podrá ser utilizada en cualquier momento y producirá el cierre de la aplicación. En caso de que un usuario se encuentre autenticado se producirá primero su desconexión y a continuación el cierre de esta. Podremos ver el menú en la siguiente imagen donde aparte del nombre de las funciones podremos ver el atajo correspondiente para manejarlas con el teclado (Salir Ctrl+S, desconectar Ctrl+D).



41Barra de tareas: Archivo

El siguiente menú, funciones, se encuentran todas las funciones que se pueden realizar cuando un usuario se encuentra en la sala, por consiguiente sus funciones solo estarán permitidas si el usuario se encuentra en esta ventana. Las funciones contenidas son: abrir mesa, ver reglas, ver mesas disponibles y ver ranking. A continuación solo se van a mostrar una imagen donde podemos ver desplegado el menú y los atajos correspondientes a estas funciones, que son:

- Abrir mesa : Ctrl+V.
- Ver reglas: Ctrl+R.
- Ver mesas: Ctrl+M.
- Ver ranking: Ctrl+W.

Desarrollo de una aplicación que permita jugar, en tiempo real, por internet al Tute



42Barra de tareas: Funciones

En el último menú se encuentra la opción de configurar la dirección donde se encuentra el servidor, es decir colocar la dirección IP de éste. Por defecto contiene la dirección de localhost “127.0.0.1” que en caso que el servidor esté en otra máquina deberá ser modificada por la dirección IP de esa. Este menú solo será visible cuando un usuario aun no se haya autenticado, a continuación se verá una imagen.



43Barra de tareas: Conexion.

9. Instalación del software

Explicación de pasos a seguir en la instalación del software necesario para la perfecta ejecución de las aplicaciones.

9.1. Instalación de la maquina virtual de java

Para la ejecución de las diferentes aplicaciones, tanto del servidor como del cliente, hace falta tener instalado la maquina virtual de java.

Los pasos a seguir son los siguientes:

1. Entrar en <http://www.java.com/es/download/manual.jsp> y descargar la versión adecuada al sistema operativo que sea utilizado.
2. Instalar la maquina virtual siguiendo los pasos recomendados en el link anterior.

Simplemente con estos pasos la aplicación cliente ya sería posible ejecutarla.

9.2. Instalación del MySql

Para la ejecución del servidor hace falta la instalación del servidor de base de datos MySql, que consta de una versión totalmente gratuita.

Los pasos a seguir son los siguientes:

1. Descargar el archivo <http://dev.mysql.com/downloads/> . Para llevar a cabo esta acción debes registrarte.
2. Comenzar la instalación en modo personalizado.
3. La configuración la podréis realizar la a vuestro gusto, solo prestar atención a dejar el puerto de conexión que se considera por defecto y marcar la casilla de crear excepción en firewall.
4. En el siguiente paso deberéis modificar la contraseña que viene por defecto para el usuario root y poner “tutetute”.
5. Darle a “execute” y comprobar que todo funciona correctamente, si no es así, posiblemente el error sea provocado por el firewall del antivirus que tengáis instalado, en el cual deberéis crear una excepción.
6. Cuando funcione todo perfectamente deberéis ejecutar el cliente por consola y crear una base de datos con el nombre “tute” con la siguiente línea de comandos “CREATE DATABASE tute;”.
7. A continuación conectaros a ella con el comando “connect tute;”.

8. Y por último deberéis cargar la base de datos ya creada en el archivo baseDatos.sql. Para ello como ya se está conectado a la base de datos ponemos el comando “source /ruta/baseDatos.sql”, así cargaríais toda la base de datos más la información contenida en ella con la que he realizado las pruebas. En caso de no querer así podréis crear las tablas manualmente poniendo las secuencias expuestas en el correspondiente apéndice.

Ahora ya se podría ejecutar la aplicación servidor teniendo cuidado que se encuentre situado en el mismo directorio que la carpeta “lib”, si no es así debéis modificar el archivo “MANIFEST.MF” incluido dentro del .jar y cambiar el path de las librerías.

10. Conclusiones y líneas futuras

Las conclusiones del presente trabajo son haber conseguido los objetivos inicialmente propuestos:

- La posibilidad de realizar una partida al tute en la que participan cuatro usuarios.
- Permitir que se jueguen varias partidas a la vez.
- Que uno o varios usuarios pueda entrar como observador en una partida.
- Llevar un registro de los usuarios que acceden a la aplicación.
- La posibilidad de que los usuarios puedan competir en un ranking, por el cual fue necesario guardar los datos de las partidas realizadas.
- Permitir chatear en dos chats diferentes, el chat de lobby y el chat de mesa.
- Un usuario puede crear una mesa privada en la cual pueda invitar a los usuarios que él desee y situarlos en el asiento que él quiera.
- La posibilidad de consultar la reglas del juego del tute.
- La encriptación de la contraseña utilizada por los usuarios.

Como líneas futuras podemos poner tres objetivos inicialmente propuestos que no fueron implementados como son:

- Un chat individual.
- Una página web para poder descargar la aplicación cliente.
- Que un usuario pueda agregar a otros usuarios como amigos o enemigos.

Otras líneas futuras a seguir pueden ser:

Desarrollo de una aplicación que permita jugar, en tiempo real, por internet al Tute

- La posibilidad de crear torneos.
- Crear una interfaz para poder jugar desde una página web.
- Diferentes tipos de salas, con otros tipos de tute, como pueden ser el tute cabrón, el subastado, etc.

11. Bibliografía y enlaces

11.1. Bibliografía

- Craig Larman. UML y Patrones: Una Introducción al Análisis y al Diseño Orientado a Objetos y al Proceso Unificado. Pearson - Prentice Hall. 2^aEdición. 2003.
- Elmasri, R y Navathe, S.B. Fundamentos de Sistemas de Bases de Datos. Addison-Wesley. 3^o edición, 2002

11.2. Enlaces de consulta y descargas

- <http://java.sun.com/j2se/1.5.0/docs/api/>
- <http://www.elrincondelprogramador.com/>
- <http://es.wikipedia.org/>
- <http://www.netbeans.org/>
- <http://www.mysql.com/>
- <http://www.lawebdelprogramador.com/>

12. Apéndices

Vamos a tocar temas referentes al proyecto que no serían adecuados meterlos dentro del cuerpo del proyecto.

12.1.Implementación física de la base de datos

Vamos a exponer la implementación física de las tablas de la base de datos, y también la consulta realizada para la obtención del ranking.

Comenzaremos con la tabla de usuario, que representa los datos guardados de cada uno de ellos:

```
Create table usuario (
    nombre      VARCHAR(20) NOT NULL,
    password   VARCHAR(80) NOT NULL,
    email      VARCHAR(256) NOT NULL,
    puntos     Integer NOT NULL,
    CONSTRAINT c_usu PRIMARY KEY(nombre)
);
```

La siguiente representada es la tabla partidaJugada, que representa cada una de las partidas que fueron realizadas en el servidor y que fueron finalizadas, mostrando los diferentes datos:

```
Create table partidaJugada (
    fechaInicio VARCHAR(80) NOT NULL,
    horaInicio  VARCHAR(80) NOT NULL,
    numMesa     Integer NOT NULL,
    puntosP1   Integer NOT NULL,
    puntosP2   Integer NOT NULL,
    CONSTRAINT c_parJ PRIMARY KEY(fechaInicio, horaInicio,
    numMesa)
);
```

Ahora es el turno de la implementación física de la tabla partidaJugador, que representa que un usuario estuvo jugando en una determinada partida y muestra si fue ganada o perdida:

```
Create table partidaJugador(  
    nombre      VARCHAR(20) NOT NULL,  
    fechaInicio VARCHAR(80) NOT NULL,  
    horaInicio  VARCHAR(80) NOT NULL,  
    numMesa     Integer NOT NULL,  
    resultado   VARCHAR(80) NOT NULL CHECK (resultado =  
        'ganada' or resultado = 'perdida'),  
    CONSTRAINT c_pardor PRIMARY KEY(nombre, fechaInicio,  
        horaInicio, numMesa),  
    CONSTRAINT c_fpar FOREIGN KEY (fechaInicio, horaInicio,  
        numMesa) REFERENCES partidaJugada(fechaInicio, horaInicio,  
        numMesa),  
    CONSTRAINT c_fusu FOREIGN KEY (nombre) REFERENCES  
        usuario(nombre)  
);
```

Por último vamos a mostrar la implementación física que en este caso no está utilizada por nuestra aplicación pero sí que haría falta para implementar una línea futura que fue analizada y diseñada pero no implementada como es que un usuario pueda seleccionar a otro usuario como amigo o enemigo.

```
Create table conocido(
    nombreUsuario  VARCHAR(20) NOT NULL,
    nombreConocido VARCHAR(20) NOT NULL,
    tipo          VARCHAR(20) NOT NULL CHECK (tipo='amigo' or
    tipo='enemigo'),
    CONSTRAINT      c_con      PRIMARY      KEY(nombreUsuario,
    nombreConocido),
    CONSTRAINT      c_fnU      FOREIGN      KEY      (nombreUsuario)
REFERENCES usuario(nombre),
    CONSTRAINT      c_fnC      FOREIGN      KEY      (nombreConocido)
REFERENCES usuario(nombre)
);
```

Ahora vamos a mostrar la consulta que se realiza para obtener el ranking.

```
SELECT usuario.nombre, usuario.puntos ,
    SUM( if(partidaJugador.resultado = \ganada\', 1, 0)) as ganadas,
    SUM( if(partidaJugador.resultado = \perdida\', 1, 0)) as perdidas
FROM usuario left join partidaJugador
    on usuario.nombre = partidaJugador.nombre
GROUP BY usuario.nombre
ORDER BY usuario.puntos DESC, usuario.nombre asc;
```

12.2.Algoritmo SHA-1

La familia SHA (Secure Hash Algorithm) es un sistema de funciones hash criptográficas relacionadas de la Agencia de Seguridad Nacional de los Estados Unidos y publicadas por el National Institute of Standards and Technology (NIST). El primer miembro de la familia fue publicado en 1993 es oficialmente llamado SHA. Sin embargo, hoy día, no oficialmente se le llama SHA-0 para evitar confusiones con sus sucesores. Dos años más tarde el primer sucesor de SHA fue publicado con el nombre de SHA-1. Existen cuatro variantes más que se han publicado desde entonces cuyas diferencias se basan en un diseño algo modificado y rangos de salida incrementados: SHA-224, SHA-256, SHA-384, y SHA-512 (llamándose SHA-2 a todos ellos).

SHA-1 es una forma de función de hash como se nombró anteriormente. Funciones que se caracterizan por dos propiedades. La primera es que son de sentido único. Esto significa que usted puede tomar un mensaje y calcular un valor de hash, pero no se puede tomar un valor hash y recrear el mensaje original. También es libre de colisión y, por tanto, no hay dos mensajes hash con el mismo valor.

SHA-1 producen una salida resumen de 160 bits (20 bytes) de un mensaje que puede tener un tamaño máximo de 264 bits, y se basa en principios similares a los usados por el profesor Ronald L. Rivest del MIT en el diseño de los algoritmos de resumen de mensaje MD4 y MD5.

La codificación hash vacía para SHA-1 corresponde a:

$$\text{SHA1}("") = da39a3ee5e6b4b0d3255bfef95601890afd80709$$

Al computar un mensaje, SHA-1 procesos de bloques de 512 bits. La longitud total del mensaje será un múltiplo de 512. Este proceso se conoce como relleno del mensaje.

SHA-1 se diferencia de SHA-0 por una sola rotación bitwise en el mensaje de su calendario de compresión función.

12.3. Tecnologías empleadas

- **JDBC:** Es una Api de Java utilizada comúnmente para conectar y manejar información, desde un programa del usuario, de una base de datos (JDBC, Java Database Connectivity), una hoja de cálculo o de archivos planos.

La conexión se realiza de forma que no importa que software de administración o manejo de la base de datos se utilice para controlarla. Desde hace tiempo las diferentes compañías productoras de un servidor de base de datos vienen utilizando ya un estándar llamado conectividad abierta a la base de datos ODBC (Open Database Connectivity). Este estándar está programado en lenguaje C, esto implica que necesiten ser instaladas previamente en el sistema correspondiente su versión compatible. Las estructuras de JDBC están construidas en esta característica, e incrementa el nivel de abstracción necesario para no perder la portabilidad que ofrece Java. Esta portabilidad se gana a través de unos clase intermedia ofrecida por cada productor, que implementa la interfaz Driver.

12.4. Herramientas utilizadas.

11.4.1. Plataforma Java

Este entorno de computación fue creado por Sun Microsystems. Esta plataforma es capaz de ejecutar aplicaciones escritas en lenguaje Java o cualquier otro lenguaje que sean compilados en bytecode, y su característica principal es que no consiste en un hardware específico o un

sistema operativo sino que es la unión de dos componentes una Maquina Virtual y un conjunto de librerías estándar.

Maquina Virtual: Es un procesador virtual, capaz de ejecutar el código resultante después de la compilación, el llamado bytecode, traduciéndolo a instrucciones nativas de la plataforma destino.

Librerías Java: Estas librerías de Java son diferentes al resto, ya que como Java está pensando en ser independiente al sistema operativo subyacente. Las librerías de Java tienen tres propósitos dentro de la Plataforma Java. Al igual que otras librerías estándar, ofrecen al programador un conjunto bien definido de funciones para realizar tareas comunes, como manejar listas de elementos u operar de forma sofisticada sobre cadenas de caracteres. Además, las librerías proporcionan una interfaz abstracta para tareas que son altamente dependientes del hardware de la plataforma destino y de su sistema operativo

Esto permite que una misma aplicación Java pueda ser ejecutada en una variedad muy grande de sistemas, sin importar el sistema operativo ni el hardware, simplemente con que estos contengan instalada la plataforma Java adecuada para ellos.

11.4.2. NetBeans

Se refiere a una plataforma para el desarrollo de aplicaciones de escritorio usando Java y a un entorno de desarrollo integrado (IDE) usando la Plataforma NetBeans.



44 NetBeans v6.7

La Plataforma NetBeans es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio grandes. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones.

La plataforma ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación. Entre las características de la plataforma están:

- Administración de las interfaces de usuario (ej. menús y barras de herramientas).
- Administración de las configuraciones del usuario.
- Administración del almacenamiento (guardando y cargando cualquier tipo de dato).
- Administración de ventanas.

- Framework basado en asistentes (diálogos paso a paso).

El IDE NetBeans es un IDE (una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas) está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender este IDE . Es un producto libre y gratuito sin restricciones de uso.

11.4.3. MagicDraw UML

UML MagicDraw es una herramienta CASE desenvuelta por No Magic, Inc. Fue implementada totalmente en lenguaje Java y está pensada para ser ejecutada en la mayor parte das plataformas



45 Magic Draw

Diseñado para analistas de negocio, analistas de software, para programadores, para ingenieros de QA (Control de Calidad), y también

para los encargados de documentar la aplicación, esta herramienta de desarrollo, dinámica y versátil, facilita el análisis y diseño de los sistemas orientados a objetos y a sus bases de datos.

Las herramientas CASE (Computer Aided Software Engineering) son diversas aplicaciones informáticas cuyo objetivo es aumentar la productividad en el desarrollo de software, reduciendo el coste de las mismas en términos de tiempo y de dinero. Las herramientas CASE nos ayudan en todas las partes que suponen un ciclo de vida de desarrollo de software, como son: diseñar el proyecto, calcular costes, implementar parte del código de forma automática con el diseño dado, compilar automáticamente, documentar o detectar errores, entre otras.

11.4.4. MySql

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. MySQL AB desarrolla MySQL como software libre en un esquema de licenciamiento dual.



46 MySql

Inicialmente, MySQL carecía de elementos considerados esenciales en las bases de datos relacionales, tales como integridad referencial y transacciones. A pesar de ello, atrajo a los desarrolladores de páginas web con contenido dinámico, justamente por su simplicidad.

Poco a poco los elementos de los que carecía MySQL están siendo incorporados tanto por desarrollos internos, como por desarrolladores de software libre. Entre las características disponibles en las últimas versiones se puede destacar:

- Amplio subconjunto del lenguaje SQL. Algunas extensiones son incluidas igualmente.
- Disponibilidad en gran cantidad de plataformas y sistemas.
- Diferentes opciones de almacenamiento según si se desea velocidad en las operaciones o el mayor número de operaciones disponibles.
- Transacciones y claves foráneas.

- Conectividad segura.
- Replicación.
- Búsqueda e indexación de campos de texto.

MySQL es muy utilizado en aplicaciones web, como Drupal o phpBB, en plataformas (Linux/Windows-Apache-MySQL-PHP/Perl/Python), y por herramientas de seguimiento de errores como Bugzilla. Su popularidad como aplicación web está muy ligada a PHP, que a menudo aparece en combinación con MySQL. MySQL es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones.

11.4.3. Otras herramientas:

- **Photoshop**

Es una herramienta de tratamiento de fotografía digital, desarrollada por Adobe Systems, inicialmente para computadores Apple pero posteriormente también para plataformas PC con sistema operativo Windows.

12.5. Lenguajes empleados

12.5.1. Java

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Las aplicaciones Java están típicamente compiladas en un *bytecode*, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.

La implementación original y de referencia del compilador, la máquina virtual y las bibliotecas de clases de Java fueron desarrollados por Sun Microsystems en 1995. Desde entonces, Sun ha controlado las especificaciones, el desarrollo y evolución del lenguaje a través del Java Community Process, si bien otros han desarrollado también implementaciones alternativas de estas tecnologías de Sun, algunas incluso bajo licencias de software libre.

El lenguaje Java se creó con cinco objetivos principales:

- A. Debería usar la metodología de la programación orientada a objetos.
- B. Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos.
- C. Debería incluir por defecto soporte para trabajo en red.

- D. Debería diseñarse para ejecutar código en sistemas remotos de forma segura.
- E. Debería ser fácil de usar.

12.5.2. SQL

El SQL es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales permitiendo gran variedad de operaciones en éstos últimos.

Es un lenguaje declarativo de "alto nivel" o "de no procedimiento", que gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros, y no a registros individuales, permite una alta productividad en codificación y la orientación a objetos. De esta forma una sola sentencia puede equivaler a uno o más programas que utilizas en un lenguaje de bajo nivel orientado a registro.

E los lenguajes de acceso a bases de datos de alto nivel, el SQL es un lenguaje declarativo. O sea, que especifica qué es lo que se quiere y no cómo conseguirlo, por lo que una sentencia no establece explícitamente un orden de ejecución.

12.5.3. UML

Lenguaje Unificado de Modelado (UML, *Unified Modeling Language*) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir

un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

Es importante resaltar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo.

Se puede aplicar en el desarrollo de software entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software, pero no especifica en sí mismo qué metodología o proceso usar.

UML no puede compararse con la programación estructurada, pues UML significa Lenguaje Unificado de Modelado, no es programación, solo se diagrama la realidad de una utilización en un requerimiento. Mientras que, programación estructurada, es una forma de programar como lo es la orientación a objetos, sin embargo, la programación orientada a objetos viene siendo un complemento perfecto de UML, pero no por eso se toma UML sólo para lenguajes orientados a objetos.

UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas.

1. Los Diagramas de Estructura enfatizan en los elementos que deben existir en el sistema modelado:
 - Diagrama de clases

- Diagrama de componentes
 - Diagrama de objetos
 - Diagrama de estructura compuesta
 - Diagrama de despliegue
 - Diagrama de paquetes
2. Los Diagramas de Comportamiento enfatizan en lo que debe suceder en el sistema modelado:
- Diagrama de actividades
 - Diagrama de casos de uso
 - Diagrama de estados
3. Los Diagramas de Interacción son un subtipo de diagramas de comportamiento, que enfatiza sobre el flujo de control y de datos entre los elementos del sistema modelado:
- Diagrama de secuencia
 - Diagrama de comunicación, que es una versión simplificada del Diagrama de colaboración
 - Diagrama de tiempos
 - Diagrama global de interacciones o Diagrama de vista de interacción

12.6. Proceso unificado (RUP)

El Proceso Unificado o RUP (Rational Unified Process) es una metodología que se caracteriza por ser dirigida por los casos de uso, centrado en la arquitectura, realización de enfoque de riesgos, iterativo e incremental.

- **Dirigida por los casos de uso:** Es un sistema de software que se crea para servir al usuario. Entonces es imprescindible saber las necesidades sus necesidades. Con usuario nos referimos no solo a personas humanas, sino también a otros sistemas.

Un caso de uso es una pieza funcional en el sistema que devuelve un valor al usuario, es decir, los casos de uso capturan los requerimientos funcionales. Todos los casos de uso unidos forman el Modelo de Casos de Uso, y contienen toda la funcionalidad que debe contener el sistema final.

Estos casos de uso van a dirigir tanto el diseño, implementación y las pruebas.

- **Centrado en la arquitectura:** El proceso unificado asume que existen diferentes modelos que cubran todos los aspectos del sistema. Por dicho motivo existen diferentes modelos y vistas que definen la arquitectura de software de un sistema.

La arquitectura surge de las necesidades de la empresa, tal y como las interpretan los usuarios, y tal y como están reflejadas en los casos de uso. Sin embargo, también está influenciada por muchos otros factores, tales como la plataforma de software en la que se ejecutará, la disponibilidad de componentes reutilizables, consideraciones de instalación, sistemas legados, requerimientos no funcionales. La arquitectura es la vista del diseño completo con las características más importantes hechas más

visibles y dejando los detalles de lado. El proceso ayuda al grupo de desarrollo a enfocarse en las metas correctas, tales como claridad, flexibilidad en los cambios futuros y rehuso.

Por consiguiente, los casos de uso deben, cuando son realizados, acomodarse en la arquitectura. Por otra parte, la arquitectura debe proveer espacio para la realización de todos los casos de uso, hoy y en el futuro.

- **Iterativo e incremental:** Desarrollar un producto de software comercial es una tarea enorme que puede continuar por varios meses o años. Es práctico dividir el trabajo en pequeños pedazos o mini-proyectos. Cada mini-proyecto es una iteración que finaliza en un incremento. Las iteraciones se refieren a pasos en el flujo de trabajo, los incrementos se refieren a crecimiento en el producto. Para ser más efectivo, las iteraciones deben estar controladas, esto es, deben ser seleccionadas y llevadas a cabo de una manera planeada.

Los desarrolladores basan su selección de qué van a implementar en una iteración en dos factores. Primero, la iteración trata con un grupo de casos de uso que en conjunto extienden la usabilidad del producto. Segundo, la iteración trata con los riesgos más importantes. Las iteraciones sucesivas construyen los artefactos del desarrollo a partir del estado en el que fueron dejados en la iteración anterior.

En cada iteración, los desarrolladores identifican y especifican los casos de uso relevantes, crean el diseño usando la arquitectura como guía, implementan el diseño en componentes y verifican que los componentes satisfacen los casos de uso. Si una iteración cumple sus metas – y

usualmente lo hace – el desarrollo continúa con la siguiente iteración. Cuando la iteración no cumple con sus metas, los desarrolladores deben revisar sus decisiones previas y probar un nuevo enfoque.

- **Enfoque de riesgos:** El proceso unificado se centra en la búsqueda riesgos críticos en etapas tempranas. El desarrollo de las iteraciones debe asegurar que siguen un orden en el cual los riesgos principales se abordan primero.

Esta metodología se divide en cuatro fases, en las cuales dentro de cada una se realizan un número de interacciones variable según el proyecto y están dirigidas a las distintas actividades:

- **Fase de Inicio:** se da una justificación a la realización del proyecto, contextualizarlo, esbozar los casos de uso e identificar los riesgos.
- **Fase de elaboración:** su actividad principal es capturar la mayoría requisitos del sistema, identificar los riesgos, y establecer y validar la arquitectura del sistema.
- **Fase de construcción:** el sistema es construido según lo especificado en la fase de elaboración, en la cual las iteraciones deben ser cortas y el resultado de cada una debe ser una parte ejecutable de software.
- **Fase de transacción:** se realiza el despliegue hacia los usuarios finales, y recoger datos para un refinamiento en iteraciones posteriores.

12.7.Patrón MVC (modelo-vista-controlador)

Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista.

- **Modelo:** Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos; por ejemplo, no permitiendo comparar un número de unidades negativo, calculando si hoy es el cumpleaños del usuario o los totales, impuestos o importes en un carrito de la compra.
- **Vista:** Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- **Controlador:** Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

Muchos sistemas informáticos utilizan un Sistema de Gestión de Base de Datos para gestionar los datos: en MVC corresponde al modelo. La unión entre capa de presentación y capa de negocio conocido en el paradigma de la Programación por capas representaría la integración entre Vista y su correspondiente Controlador de eventos y acceso a datos, MVC no pretende discriminar entre capa de negocio de capa de presentación pero si pretende separar la capa visual gráfica de su correspondiente programación y acceso a datos algo que mejora el desarrollo y mantenimiento de la Vista y el Controlador en paralelo ya que ambos cumplen ciclos de vida muy distintos entre si.

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo que sigue el control generalmente es el siguiente:

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.)

2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega.

3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario. Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo. El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, el patrón de observador puede ser utilizado para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio a la vista aunque puede dar la orden a la vista para que se actualice.

5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.