

Universidad Autónoma del Estado de México

Unidad Académica Profesional Tianguistenco

Ingeniería en Software

Unidad de Aprendizaje:

Sistemas Distribuidos

Profesor:

Jovani Armeaga García

Titulo:

Proyecto: Juego de Ahorcado

López Palma Carlos Jonathan ...... N.C: 1413313

David Velázquez Ramírez ............. N.C: 1771615

Uriel García Martínez ……………... N.C: 1416308

Edwin Misael Vázquez Rivera …… N.C: 1414972

César Pedraza Hernández …......... N.C: 1774280

Contenido

[Introducción 4](#_Toc57917372)

[Antecedentes 5](#_Toc57917373)

[Historia del juego del ahorcado 5](#_Toc57917374)

[Videojuegos y sistemas distribuidos 5](#_Toc57917375)

[Planteamiento del problema 6](#_Toc57917376)

[Objetivos generales 6](#_Toc57917377)

[Objetivos específicos 6](#_Toc57917378)

[Hipótesis 7](#_Toc57917379)

[Justificación 8](#_Toc57917380)

[Delimitación del problema 9](#_Toc57917381)

[Definición de requerimientos 9](#_Toc57917382)

[Requerimientos Funcionales 10](#_Toc57917383)

[Requerimientos No Funcionales 16](#_Toc57917384)

[Arquitectura SD 22](#_Toc57917385)

[Diseño 23](#_Toc57917386)

[Prototipo 23](#_Toc57917387)

[Diagrama de casos de uso 24](#_Toc57917388)

[Diagrama de secuencias 25](#_Toc57917389)

[Diagrama de bloques 27](#_Toc57917390)

[Diagrama entidad-relación 27](#_Toc57917391)

[Diccionario de datos 28](#_Toc57917392)

[Tablas relacionales 28](#_Toc57917393)

[Descripción de tablas 29](#_Toc57917394)

[Interfaces 30](#_Toc57917395)

[Desarrollo 31](#_Toc57917396)

[Experimentación 35](#_Toc57917397)

[Resultados 38](#_Toc57917398)

[Conclusiones 39](#_Toc57917399)

**Índice de Ilustraciones**

[1. Arquitectura Cliente-Servidor 22](#_Toc57917400)

[2. Prototipo pantalla principal 23](#_Toc57917401)

[3. Diagrama de casos de uso A 23](#_Toc57917402)

[4. Diagrama de casos de uso A 24](#_Toc57917403)

[5. Diagrama de casos de uso B 24](#_Toc57917404)

[6. Diagrama de secuencias A 25](#_Toc57917405)

[7. Diagrama de secuencias B 26](#_Toc57917406)

[8. Diagrama de bloques. 27](#_Toc57917407)

[9. Diagrama Entidad-Relación 27](#_Toc57917408)

[10. Tablas relacionales 28](#_Toc57917409)

[11. Pantalla principal 30](#_Toc57917410)

[12. Pantalla partida 30](#_Toc57917411)

[13. Desarrollo A1 31](#_Toc57917412)

[14. Desarrollo A2 32](#_Toc57917413)

[15. Desarrollo B1 32](#_Toc57917414)

[16. Desarrollo C1 33](#_Toc57917415)

[17. Desarrollo C2 33](#_Toc57917416)

[18. Desarrollo D1 34](#_Toc57917417)

[19. Desarrollo D2 34](#_Toc57917418)

[20. Error recibir mensaje 1 35](#_Toc57917419)

[21. Corrección. Mensaje recibido 35](#_Toc57917420)

[22. Error esperando mensaje 36](#_Toc57917421)

[23. Error servidor envía mensaje 36](#_Toc57917422)

[24. Servidor acepta clientes 36](#_Toc57917423)

[25. Error conexión inesperada 37](#_Toc57917424)

**Índice de Tablas**

[Tabla 1. Diccionario de datos. Palabras 28](#_Toc57917432)

[Tabla 2. Diccionario de datos. Partida 28](#_Toc57917433)

[Tabla 3. Diccionario de datos. Rondas 28](#_Toc57917434)

[Tabla 4. Descripción de tablas. Palabras 29](#_Toc57917435)

[Tabla 5. Descripción de tablas. Partida 29](#_Toc57917436)

[Tabla 6. Descripción de tablas. Rondas 29](#_Toc57917437)

# **Introducción**

Esta aplicación/juego es desarrollada para la materia de Sistemas Distribuidos de la carrera de Ingeniería de Software, con la finalidad de aplicar temas abordados en las diferentes unidades de dicha materia. Se plantea hacerlo mediante el desarrollo del juego ahorcado. El ahorcado (también llamado colgado) es un juego de lápiz y papel, en el que el objetivo es adivinar una palabra o frase. En un panel esconde una frase o palabra secreta que deberá de ser averiguada por los jugadores. Sobre la frase se dará una pista y se podrán ir descubriendo las letras de la misma. Tradicionalmente el juego consta de dos jugadores, uno de ellos elige la palabra y el otro trata de adivinarla. Se pretenden que este juego funcione como un sistema distribuido en el que al menos 5 jugadores en una plataforma intenten adivinar la palabra proporcionada al azar por el sistema.

En el siguiente Documento se pretende explicar la planeación y desarrollo del proyecto en sus diferentes etapas y/o apartados. Tal y como se plantea en la Ingeniería de Software y la Ingeniería de Requerimientos. Documentar el Programa a realizar.

# **Antecedentes**

## Historia del juego del ahorcado

Al parecer surgió en la época victoriana," dice Tony Augarde autor de "La Guía de Oxford de Juegos de palabras" (Oxford University Press).

El juego es mencionado en 1894 en "The Traditional Games of England, Scotland, and Ireland - Vol I" (Juegos tradicionales de Inglaterra, Escocia e Irlanda) ​ de Alice Bertha Gomme bajo el nombre "Birds, Beasts, and Fishes" (Pájaros, Bestias y Peces). Las reglas eran simples: un jugador anota la primera y última letra de una palabra de un animal, y el otro jugador adivina las letras en el medio.

En otras fuentes el juego se llama "Horca", "El Juego de Hangin'", o "Suspensión".

El Ahorcado ha aparecido en el sistema de videojuegos Speak & Spell de 1978 bajo el nombre de "Palabra Misterio" y en la actualidad (2014), a veces, se juega en los foros de Internet.

## Videojuegos y sistemas distribuidos

La posibilidad de utilizar un ordenador para jugar en red comenzó en torno a 1979, cuando un grupo de estudiantes de la Universidad de Essex crearon una versión informática multiusuario de un juego de rol llamado “Dungeons & Dragons”, basado en una interfaz de texto mediante el cual los usuarios podían ir avanzando en la historia. Así surgió un nuevo tipo de juegos conocidos como MUD (Multi-User Dungeons o Domains) que se desarrollaría rápidamente por la aún poco conocida Internet, surgiendo así las primeras comunidades virtuales.

El primer juego multiusuario que incorporó imágenes fue Hábitat en 1986, creado por Lucas Films Games y destinado para el Commodore 64. De él surgieron posteriormente juegos como el EverQuest, Asheron’s o Ultima Online.

Pero la verdadera revolución de los juegos en red surgió en 1993 con la creación de la World Wide Web. Los usuarios tenían la posibilidad de acceder gratuitamente a versiones reducidas de videojuegos para ordenador con fines básicamente promocionales, como es el caso de la primera entrega del Doom. Además, la rápida difusión de Internet como medio de entretenimiento facilitó la mejora de las tecnologías para la conexión en red de usuarios y su acercamiento a la sociedad.

# **Planteamiento del problema**

Se plantea la creación de un videojuego basado en el juego del ahorcado, con la finalidad de que más de dos personas puedan interactuar mediante una red informática, donde se espera podrán jugar desde distintos equipos de cómputo mediante una aplicación.

# **Objetivos generales**

Se plantea el desarrollo de una aplicación/juego basado en un sistema distribuido, para emplear técnicas de ingeniería en software, para ello se implementará la arquitectura de cliente servidor, es un modelo de diseño de software en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta y viceversa.

El juego consiste en el clásico ahorcado, o bien conocido como “Horca", "El Juego de Hangin'", o "Suspensión", El juego del Ahorcado consiste en 5 partidas, cada una con una palabra para descubrir. Cada palabra valdrá 60 puntos, que serán divididos entre todos los jugadores que descubran la palabra. Así, si un jugador acierta la palabra, ganará 60 puntos; Si dos jugadores aciertan, cada uno ganará 30 puntos, y así por delante.

# **Objetivos específicos**

**Desarrolladores:**

* Requerimientos del Software: Es importante definir de manera clara y concisa los requerimientos, para que permita llevar a cabo un desarrollo eficaz por parte de los ingenieros en software, para lograr el objetivo de implementar las técnicas de la arquitectura cliente servidor.
* Asesorías con especialistas: Con el propósito de capacitar a los desarrolladores, y con la finalidad de que adquieran más conocimientos, que ayuden en el proceso de desarrollo del sistema a realizar.
* Análisis de Requerimientos: Clasificación de información para realizar su respectivo análisis, y generar un modelo de requerimientos y poder diseñar una estructura del Sistema y poder resolver necesidades del usuario.
* Diseño de Arquitectura Cliente servidor: Implementar técnicas de software para el desarrollo del modelo cliente servidor estable, adaptada para el propósito del sistema, que le permita un correcto funcionamiento.
* Interfaz de Usuario: Diseñar una interfaz acorde a las necesidades del del modelo a implementar.
* Pruebas y Mantenimiento: Las pruebas tienen el propósito de identificar errores, fallas, u cualquier otra anomalía que interfiera en el correcto funcionamiento del producto final esperado.
* Compatibilidad del Sistema: Será necesario definir con qué plataformas y software de apoyo, se trabajará para el desarrollo del proyecto, su debida instalación y con sus respectivas versiones.

**Para Beneficio del sistema:**

* Sistema Práctico: Proporcionar un sistema que le permita a los usuarios adaptarse de manera fácil e intuitiva. Que cumpla con los criterios de usabilidad.
* Mejorar el tiempo en cada uno de los procesos de ejecución

### Hipótesis

La programación en red siempre ha sido dificultosa, el programador debía de conocer la mayoría de los detalles de la red, incluyendo el hardware utilizado, los distintos niveles en que se divide la capa de red, las librerías necesarias para programar en cada capa, etc.

Pero, la idea simplemente consiste en obtener información desde otra máquina, aportada por otra aplicación software. Por lo tanto, de cierto modo se puede reducir al mero hecho de leer y escribir archivos, con ciertas salvedades.

Es por eso por lo que se plantea utilizar la tecnología Java para comunicar diversos ordenadores y que puedan participar en un juego de Ahorcado.

# **Justificación**

Una de las características más importantes de Java es su capacidad y, a la vez, facilidad para realizar aplicaciones que funcionen en red. La mayoría de los detalles de implementación a bajo nivel están ocultos y son tratados de forma transparente por la JVM (Java Virtual Machine).

Los programas son independientes de la arquitectura y se ejecutan indistintamente en una gran variedad de equipos con diferentes tipos de microprocesadores y sistemas operativos.

En las aplicaciones en red es muy común el paradigma cliente-servidor. El servidor es el que espera las conexiones del cliente (en un lugar claramente definido) y el cliente es el que lanza las peticiones a la maquina donde se está ejecutando el servidor, y al lugar donde está esperando el servidor (el puerto(s) específico que atiende). Una vez establecida la conexión, ésta es tratada como un stream (flujo) típico de entrada/salida.

A través de las clases del paquete java.net, los programas Java pueden utilizar TCP o UDP para comunicarse a través de Internet. Las clases URL, URLConnection, Socket, y SocketServer utilizan TCP para comunicarse a través de la Red. Las clases DatagramPacket y DatagramServer utilizan UDP.

TCP proporciona un canal de comunicación fiable punto a punto, lo que utilizan para comunicarse las aplicaciones cliente-servidor en Internet. Las clases Socket y ServerSocket del paquete java.net proporcionan un canal de comunicación independiente del sistema utilizando TCP, cada una de las cuales implementa el lado del cliente y el servidor respectivamente.

Así el paquete java.net proporciona, entre otras, las siguientes clases, que son las que se verán con detalle:

* Socket: Implementa un extremo de la conexión TCP.
* ServerSocket: Se encarga de implementar el extremo Servidor de la conexión en la que se esperarán las conexiones de los clientes.
* DatagramSocket: Implementa tanto el servidor como el cliente cuando se utiliza UDP.
* DatagramPacket: Implementa un datagram packet, que se utiliza para la creación de servicios de reparto de paquetes sin conexión.
* Sistemas Distribuidos: Sockets en Java EUI-SG/INFOR.UVA.ES 3
* InetAddress: Se encarga de implementar la dirección IP.

Se utilizará el IDE Netbeans ya que puede ser usada para desarrollar cualquier tipo de aplicación, Reutilización de módulos, Permite el uso de herramientas AWT para generación de interfaces y requiere de una Instalación y actualización simple.

# **Delimitación del problema**

El videojuego se basará en el juego del ahorcado, tendrá una capacidad de 2 a 5 jugadores, se podrá ingresar mediante la aplicación y se podrá ingresar a una partida mediante un código de la sala, las palabras serán dadas por el sistema de forma aleatoria, ganara el jugador con el mejor puntaje.

# **Definición de requerimientos**

Requerimientos funcionales

* RF1: Ingresar a partida
* RF2: Sala de partida
* RF3: Iniciar partida
* RF4: Ronda
* RF5: Seleccionar palabra
* RF6: Ventana de servidor

Requerimientos no funcionales

* RNF1: Eficiencia del Programa
* RNF2: Usabilidad del Sistema
* RNF3: Dependibilidad del Sistema
* RNF4: Compatibilidad del sistema
* RNF5: Seguridad de comunicación
* RNF6: Diseño de interfaz

# **Requerimientos Funcionales**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | RF1 | | |
| **Nombre** | Ingresar a Partida | | |
| **Descripción** | El usuario debe ingresar un nombre de jugador, para entrar a una partida. | | |
| **Funcionamiento** | El usuario debe ingresar en un campo de texto (textfield) un nombre de usuario que lo identifique dentro de la sala.  El nombre debe cumplir con las siguientes características:   * Longitud mínima: 5 * Longitud máxima: 10 * El nombre puede llevar letras (a-z, A-Z), dígitos (0-9) y caracteres especiales (. (punto) - (guion), \_ (guion bajo)). * No usar caracteres que no se encuentren entre los enumerados anteriormente.   Debe ingresar la IP del servidor en un textfield para conectarse a la partida.  Botón entrar: al presionar, muestra la pantalla de sala.  Al entrar a la sala de partida, se comprueba que el nombre cumpla con las especificaciones anteriores, de lo contrario, se mantiene en la pantalla de Ingresar usuario y muestra un mensaje con el problema. Si el nombre ingresado es el mismo que el de un jugador dentro de la sala de partida, muestra un mensaje al usuario, solicitando cambiar su nombre.  Si el campo está en blanco, no podrá ingresar a la sala.  Para entrar a la sala, se debe presionar el botón “entrar”.  Cuando entra a la partida le asigna un ID al jugador. | | |
| **Prioridad de Requerimiento** | Alta | **Exigencia de requerimiento** | Exigible |
| **Relación** | RF2 | | |
| **Editor** | Carlos Jonathan López Palma | | |
| **Fecha** | 21 de octubre de 2020 | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | RF2 | | |
| **Nombre** | Sala de partida | | |
| **Descripción** | Se inicia una sala cuando un usuario entra a una partida. | | |
| **Funcionamiento** | La sala se inicia, cuando un usuario ingresa con su usuario, o se une a la partida si ya hay más usuarios dentro de la sala.  Una sala permite hasta 2 jugadores. Los jugadores dentro de la partida, se enlistan de lado superior izquierdo, en orden alfabético. Al lado derecho del nombre del jugador, se muestra su contador de puntos (Al comienzo de la partida, los puntos son 0 para todos los jugadores):   * Usuario – 0pts   La partida comienza cuando entran 2 jugadores. | | |
| **Prioridad de Requerimiento** | Alta | **Exigencia de requerimiento** | Exigible |
| **Relación** | RF1, RF3 | | |
| **Editor** | David Velázquez Ramírez | | |
| **Fecha** | 21 de octubre 2020 | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | RF3 | | |
| **Nombre** | Iniciar partida | | |
| **Descripción** | La partida comienza cunando los 2 jugadores entran a la sala. | | |
| **Funcionamiento** | Cuando haya 2 jugadores en la sala, la partida comenzara. Cada partida constara de 4 rondas para adivinar la palabra u oración.  Al entrar a la pantalla de juego, se muestran los puntos de cada jugador junto al nombre, la cantidad de puntos acumulados. Al entrar a partida, los putos de un jugador se establece en 0. | | |
| **Prioridad de Requerimiento** | Alta | **Exigencia de requerimiento** | Exigible |
| **Relación** | RF4, RF5 | | |
| **Editor** | Uriel García Martínez | | |
| **Fecha** | 21 de octubre 2020 | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | RF4 | | |
| **Nombre** | Ronda | | |
| **Descripción** | Cada partida tiene 4 rondas, en cada debe adivinar la palabra que el servidor le establezca. | | |
| **Funcionamiento** | Cada que una ronda comienza, el jugador debe adivinar la palabra enviada por el servidor. Cada jugador adivinará una palabra 4 veces (cada ronda).  Si el carácter ingresado por un jugador coincide con algún carácter de la oración o palabra, el carácter se agrega a en la posición en la que aparece.  Por cada letra que ingrese un jugador y esta no coincida con alguna de la palabra u oración a buscar, se agrega una extremidad al personaje:   * Una cabeza * Dos brazos * Dos piernas * Un torso   Si el cuerpo consigue todas sus extremidades, la roda se pierde, se le resta 1 punto al jugador que adivina. Si el jugador no tiene puntos se mantiene en 0.  Cuando un jugador completa todas las letras, la ronda termina y el jugador que adivina, gana 1 punto.  Hay un contador de 30 segundos que le avisa al jugador que adivina su tiempo restante. Si el jugador no ingresa una letra en ese tiempo, se considera como que la letra no coincide. | | |
| **Prioridad de Requerimiento** | Alta | **Exigencia de requerimiento** | Exigible |
| **Relación** | RF3, RF5 | | |
| **Editor** | Cesar Pedraza | | |
| **Fecha** | 21 de octubre 2020 | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | RF5 | | |
| **Nombre** | Seleccionar palabra | | |
| **Descripción** | Se le solicita al usuario que establece la palabra que la escriba. | | |
| **Funcionamiento** | Cuando la sala se crea, el servidor busca en la base de datos 4 palabras que enviara a cada jugador. La base de datos cuenta con un conjunto de 100 palabras y oraciones.  Cada palabra u oración cuenta con las siguientes características:   * ID: un identificador numérico para identificarlo en la lista de palabras. * CONTENIDO: la palabra u oración con las siguientes características: * Longitud: mínimo 2, máximo 20 * Utiliza caracteres (A-Z) y espacios en blanco (Los espacios no cuentan para adivinar) * TEMA: el tema al que pertenece la palabra, (METERIA, PELICULAS, OBJETOS, SERES VIVOS, etc.) * PISTA: una oración que describe una característica simple de la palabra u oración.   La pista se muestra en la parte inferior de la horca. | | |
| **Prioridad de Requerimiento** | Alta | **Exigencia de requerimiento** | Exigible |
| **Relación** | RF3, RF4 | | |
| **Editor** | Edwin Misael Vázquez Rivera | | |
| **Fecha** | 21 de octubre 2020 | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | RF6 | | |
| **Nombre** | Ventana de servidor | | |
| **Descripción** | Muestra la información del servidor, los jugadores y la partida | | |
| **Funcionamiento** | * Muestra la información de servidor (IP) * Nombres de usuarios y puntaje * Movimientos que realiza cada jugador * Mensajes enviados a cada cliente y recibidos de estos mismos. | | |
| **Prioridad de Requerimiento** | Alta | **Exigencia de requerimiento** | Exigible |
| **Relación** | RF3, RF4, RF5 | | |
| **Editor** | Edwin Misael Vázquez Rivera | | |
| **Fecha** | 21 de octubre 2020 | | |

# **Requerimientos No Funcionales**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | RNF1 | | |
| **Nombre** | Eficiencia del Programa | | |
| **Descripción** | El programa tendrá un tiempo de respuesta inferior a 30 segundos al abrir cada una de las ventanas u opciones del sistema. | | |
| **Funcionamiento** | -Permite realizar funciones en tiempo y forma  -Tiempo de respuesta corto  -Sin retardos | | |
| **Prioridad de Requerimiento** | Alta | **Exigencia de requerimiento** | Exigible |
| **Relación** |  | | |
| **Editor** | Uriel García Martínez | | |
| **Fecha** | 21 de octubre 2020 | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | RNF2 | | |
| **Nombre** | Usabilidad del Sistema | | |
| **Descripción** | Interfaces de manejo adecuado para el usuario y mensajes de error que orientados al usuario final. | | |
| **Funcionamiento** | El sistema posee interfaces diseñadas por la recomendación del cliente de mismas que son fácil de entender (según el cliente), haciendo uso de iconos y colores que faciliten la comprensión del usuario en las funciones del programa. | | |
| **Prioridad de Requerimiento** | Alta | **Exigencia de requerimiento** | Exigible |
| **Relación** |  | | |
| **Editor** | Cesar Pedraza | | |
| **Fecha** | 21 de octubre 2020 | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | RNF3 | | |
| **Nombre** | Dependibilidad del Sistema | | |
| **Descripción** | El sistema estará disponible las veces que sea requerido | | |
| **Funcionamiento** | El sistema tendrá la mayor rapidez posible por lo que también tendrá un tiempo de respuesta en caso de presentar fallas, es decir que se trabe el sistema, el tiempo de respuesta en la aplicación en caso de no reaccionar es inferior a 180 segundos. | | |
| **Prioridad de Requerimiento** | Alta | **Exigencia de requerimiento** | Exigible |
| **Relación** |  | | |
| **Editor** | Edwin Misael Vázquez Rivera | | |
| **Fecha** | 21 de octubre 2020 | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | RNF4 | | |
| **Nombre** | Compatibilidad del sistema | | |
| **Descripción** | Será compatible con computadoras personales o laptops  Versiones Windows desde la versión XP hasta Windows 10 | | |
| **Funcionamiento** | El sistema va a estar disponible para la plataforma del sistema operativo Windows.   * Windows XP * Windows Vista * Windows 7 * Windows 10   Requerirá   * Sistema Operativo Windows (cualquier versión de las incluidas anteriormente) * 1Gb de memoria RAM como mínimo * 1Gb de memoria en el equipo de cómputo (disco duro)   Nota: Si se cuenta con mejores requisitos, será mejor. | | |
| **Prioridad de Requerimiento** | Alta | **Exigencia de requerimiento** | Exigible |
| **Relación** |  | | |
| **Editor** | Carlos Jonathan López Palma | | |
| **Fecha** | 21 de octubre 2020 | | |

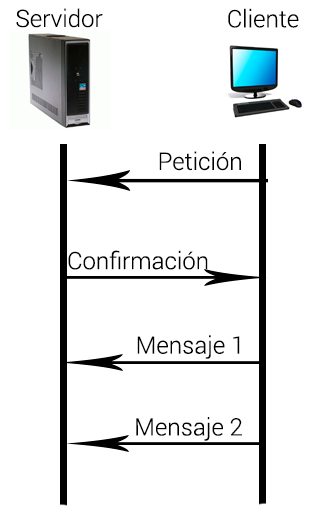
|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | RNF5 | | |
| **Nombre** | Seguridad de comunicación | | |
| **Descripción** | Sin pérdida de datos | | |
| **Funcionamiento** | El programa utiliza métodos de comunicación seguros, para prevenir la perdida de información y la filtración de esta. | | |
| **Prioridad de Requerimiento** | Alta | **Exigencia de requerimiento** | Exigible |
| **Relación** |  | | |
| **Editor** | David Velázquez Ramírez | | |
| **Fecha** | 21 de octubre 2020 | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | RNF6 | | |
| **Nombre** | Diseño de interfaz | | |
| **Descripción** | Colores de pantallas, tamaños y colores de letras | | |
| **Funcionamiento** | Letra:   * Título: Arial 36 (bold) color negro * Subtitulo: Arial 14 (bold) color negro * Textos: Arial 14 color negro   Pantallas:   * Colores:   + Amarillo paja: FEF9E7   + Verde pistacho: D5F1E3   Resolución:   * 750 ancho * 600 alto | | |
| **Prioridad de Requerimiento** | Alta | **Exigencia de requerimiento** | Exigible |
| **Relación** |  | | |
| **Editor** | David Velázquez Ramírez | | |
| **Fecha** | 21 de octubre 2020 | | |

# **Arquitectura SD**

El modelo Cliente/Servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Las aplicaciones Clientes realizan peticiones a una o varias aplicaciones Servidores, que deben encontrarse en ejecución para atender dichas demandas.

El modelo Cliente/Servidor permite diversificar el trabajo que realiza cada aplicación, de forma que los Clientes no se sobrecarguen, cosa que ocurriría si ellos mismos desempeñan las funciones que le son proporcionadas de forma directa y transparente. En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema. Tanto el Cliente como el Servidor son entidades abstractas que pueden residir en la misma máquina o en máquinas diferentes.



. Arquitectura Cliente-Servidor

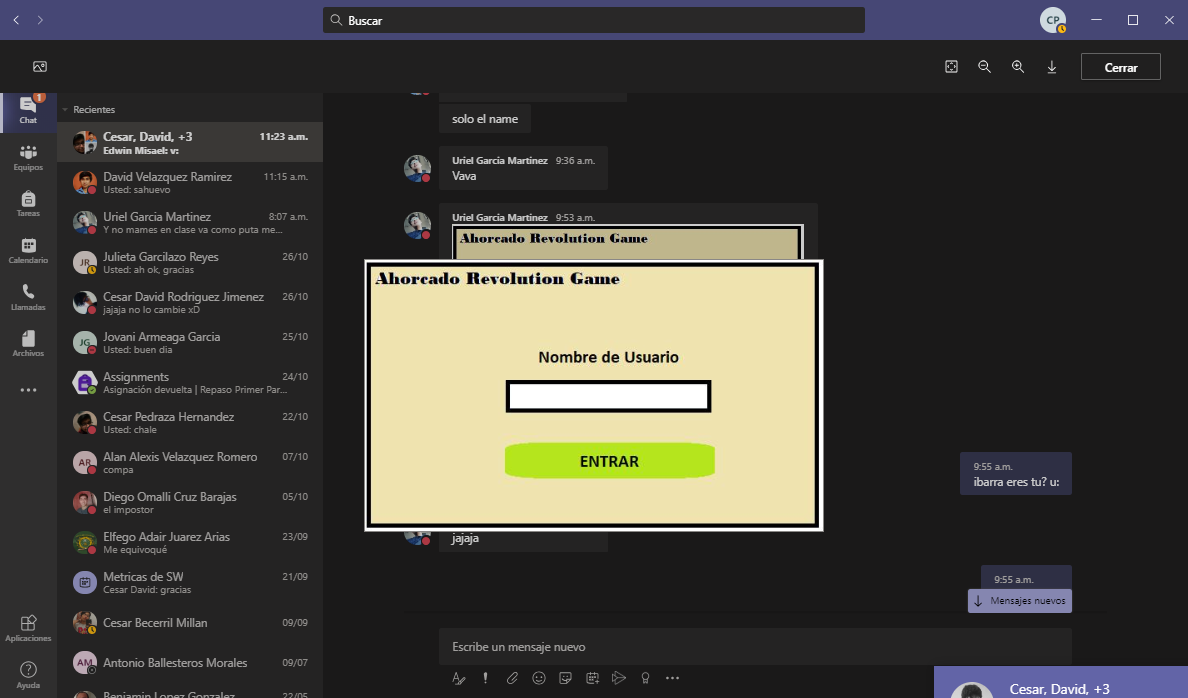
Bibliotecas de Sockets en java:

* import java.net.ServerSocket;
* import java.net.Socket;

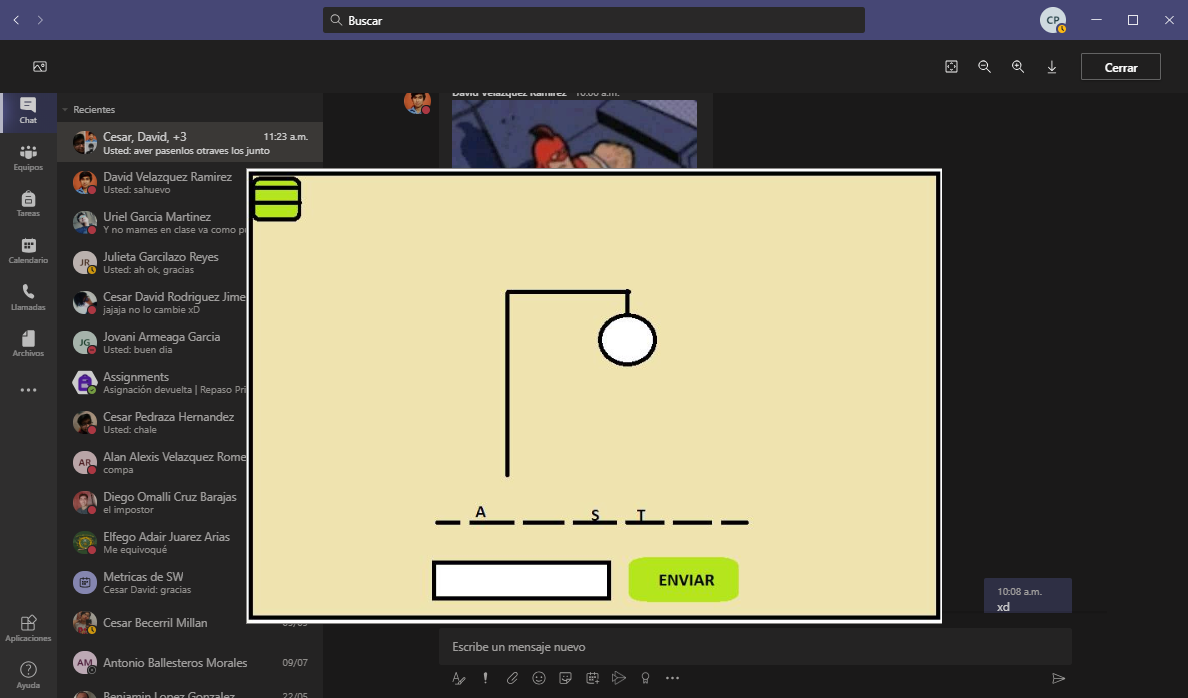
La clase Conexión simplemente nos brinda los datos que, necesitados, mensajes de entrada, flujo de salida socket para el Cliente y socket para el servidor, estos últimos respectivamente inicializados desde el constructor. La clase Servidor básicamente estará a la espera de que un cliente se conecte a él usando el socket en el puerto establecido, recibirá los mensajes, los mostrará y cerrará la conexión, es todo.

# **Diseño**

## Prototipo

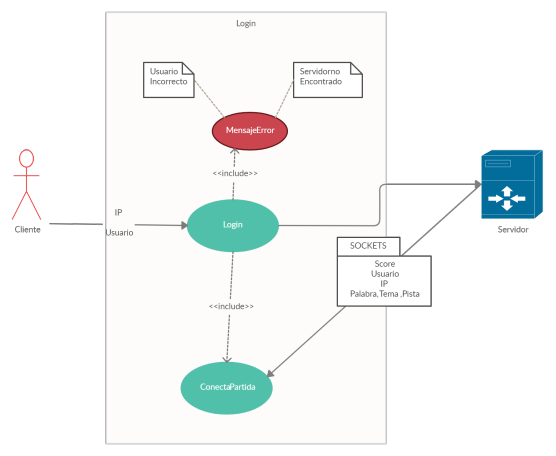


. Prototipo pantalla principal

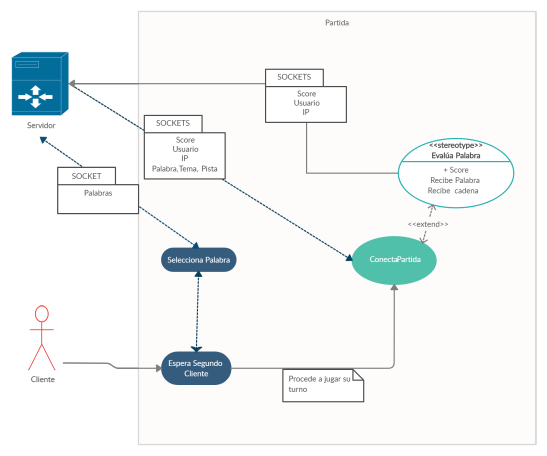


. Diagrama de casos de uso A

# **Diagrama de casos de uso**

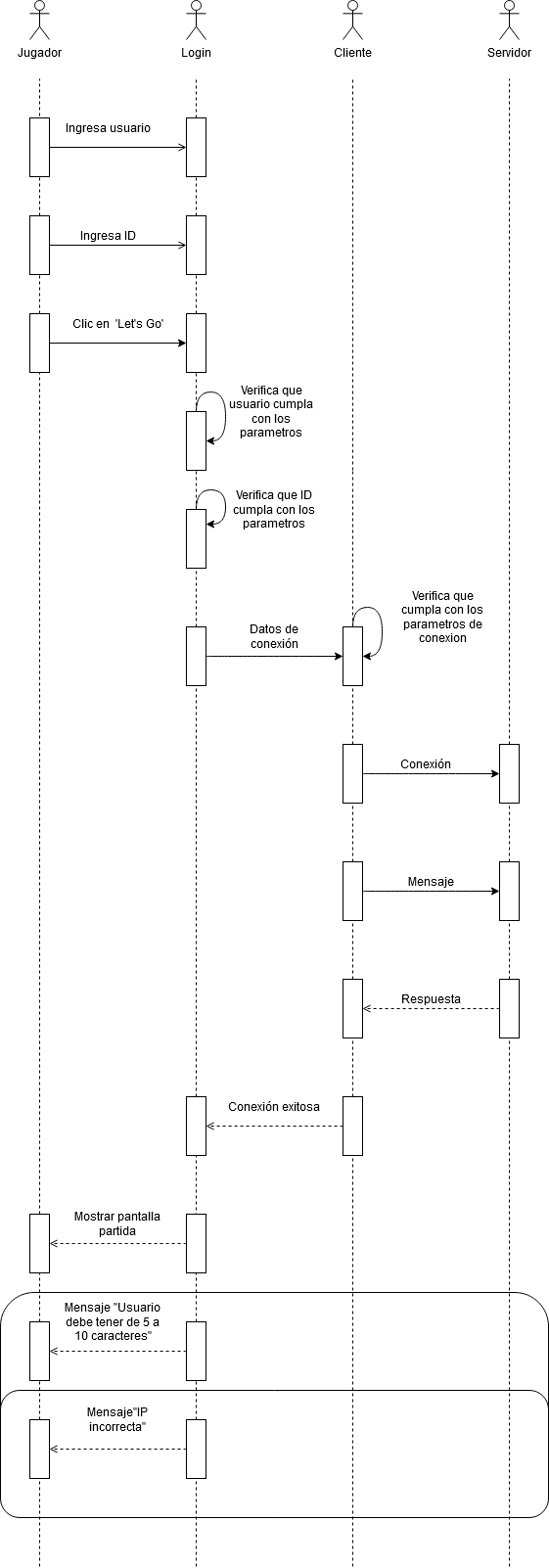


. Diagrama de casos de uso A

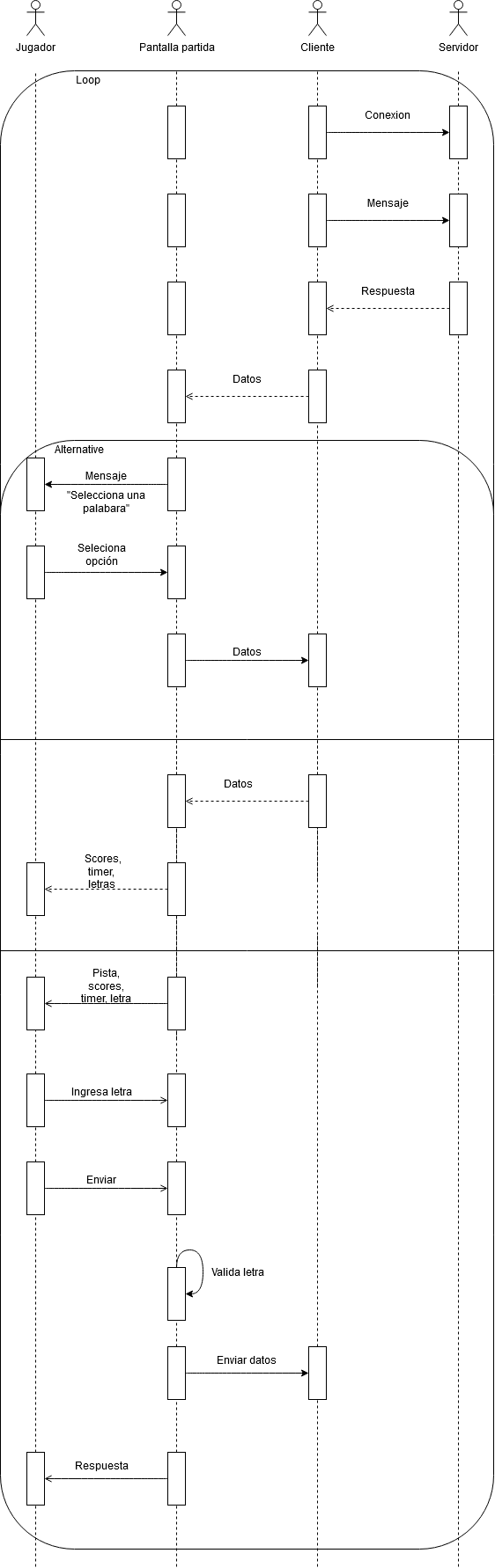


. Diagrama de casos de uso B

# **Diagrama de secuencias**

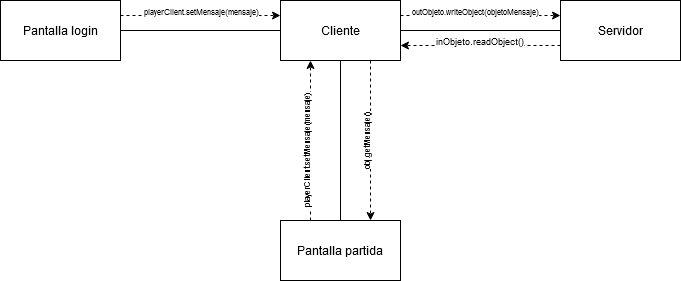


. Diagrama de secuencias A



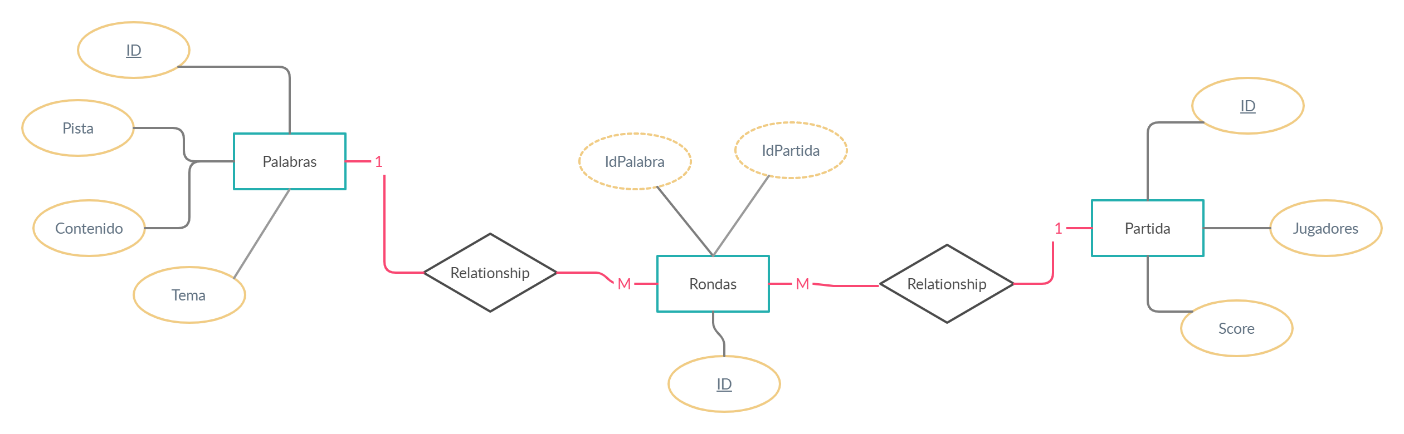
. Diagrama de secuencias B

# **Diagrama de bloques**



. Diagrama de bloques.

# **Diagrama entidad-relación**



. Diagrama Entidad-Relación

# **Diccionario de datos**

|  |  |
| --- | --- |
| **Palabras** | |
| **Campos** | **Restricciones** |
| Id | Auto-Increment, Primary Key, Not Null |
| Contenido | Varchar(20), Not Null, Min 8, Max20 |
| Tema | Varchar 20 Not Null Default ‘Sin Tema’ |
| Pista | Varchar(30) Not Null Default ‘Sin Pista’ |

Tabla . Diccionario de datos. Palabras

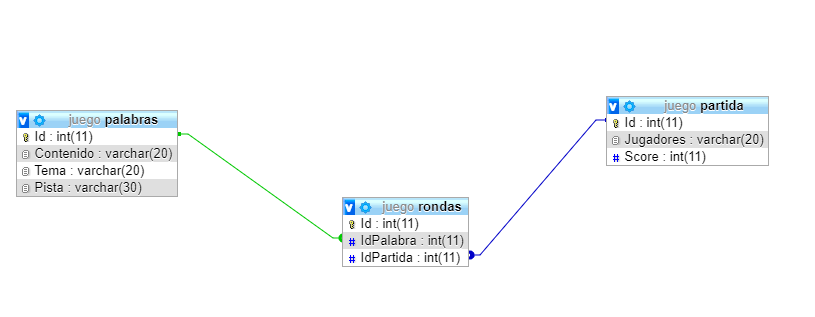
|  |  |
| --- | --- |
| **Partida** | |
| **Campos** | **Restricciones** |
| Id | Auto-Increment, Primary Key, Not Null |
| Jugadores | Varchar(20) Not Null, Min 8, Max 20 Default ‘Sin Nombre’ |
| Score | Int, Default 0, Not Null |

Tabla . Diccionario de datos. Partida

|  |  |
| --- | --- |
| **Rondas** | |
| **Campos** | **Restricciones** |
| Id | Auto-Increment, Primary Key, Not Null |
| IdPalabra | Int Forean Key Palabras |
| IdPartida | Int Forean Key Partida |

Tabla . Diccionario de datos. Rondas

# **Tablas relacionales**



. Tablas relacionales

# **Descripción de tablas**

|  |  |
| --- | --- |
| **Palabras** | |
| **Campos** | **Restricciones** |
| Id | Auto-Increment, Primary Key, Not Null |
| Contenido | Varchar(20), Not Null, Min 8, Max20 |
| Tema | Varchar 20 Not Null Default ‘Sin Tema’ |
| Pista | Varchar(30) Not Null Default ‘Sin Pista’ |

Tabla . Descripción de tablas. Palabras

La tabla **Palabras** es la tabla que contendrá al menos cien registros sobre diferentes palabras, que servirán para ser elegidas al azar dentro del sistema. Esta tabla cuenta con un identificador (PK) de tipo Auto incrementable, El campo contenido de tipo cadena, no acepta valores nulos y se debe determinar que tenga una longitud mínima de 8 caracteres y máxima de 20. Los campos Tema y Pista también son de tipo cadena, el primero con una longitud de 20 y el segundo de 30, ambos no aceptan registros vacíos o de tipo Null, y tiene designado un valor por defecto.

|  |  |
| --- | --- |
| **Partida** | |
| **Campos** | **Restricciones** |
| Id | Auto-Increment, Primary Key, Not Null |
| Jugadores | Varchar(20) Not Null, Min 8, Max 20 Default ‘Sin Nombre’ |
| Score | Int, Default 0, Not Null |

Tabla . Descripción de tablas. Partida

La tabla **Partida** es la tabla que contendrá los registros de cada partida por cada 5 jugadores, en la base de datos, esta tabla cuenta con un Identificador (PK) de tipo Auto incrementable y no aceptaba valores vacíos. El campo jugadores es un dato de tipo cadena con longitud de 20 caracteres como máximo, no acepta valores vacíos o Null, y se define como un mínimo de 8 caracteres y un máximo de 20. Por último, el campo Score es un dato de tipo entero o numérico, no acepta valores vacíos y por default tiene aginado el valor de 0.

|  |  |
| --- | --- |
| **Rondas** | |
| **Campos** | **Restricciones** |
| Id | Auto-Increment, Primary Key, Not Null |
| IdPalabra | Int Forean Key Palabras |
| IdPartida | Int Forean Key Partida |

Tabla . Descripción de tablas. Rondas

La tabla **Rondas** define la relación entre las dos tablas antes descritas, en primera instancia cuenta con un identificador (PK) de tipo auto incrementable, y tiene dos campos IdPalabra e IdPartida, son campos de tipo numérico enteros, y sus valores derivan de ser Llaves foráneas (FK), la primera es FK de la tabla palabras y la segunda es FK de la tabla partida.

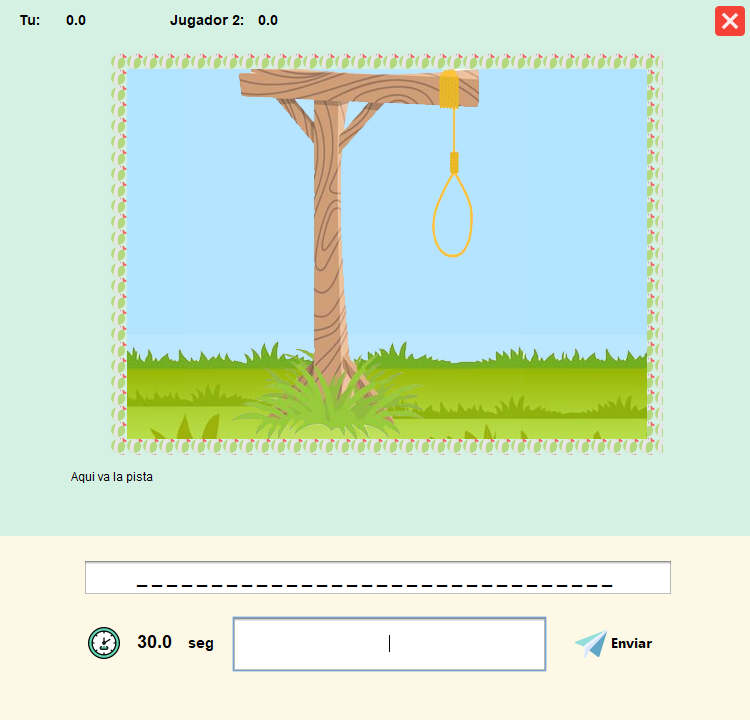
# **Interfaces**

**Pantalla principal**



. Pantalla principal

**Pantalla de partida**

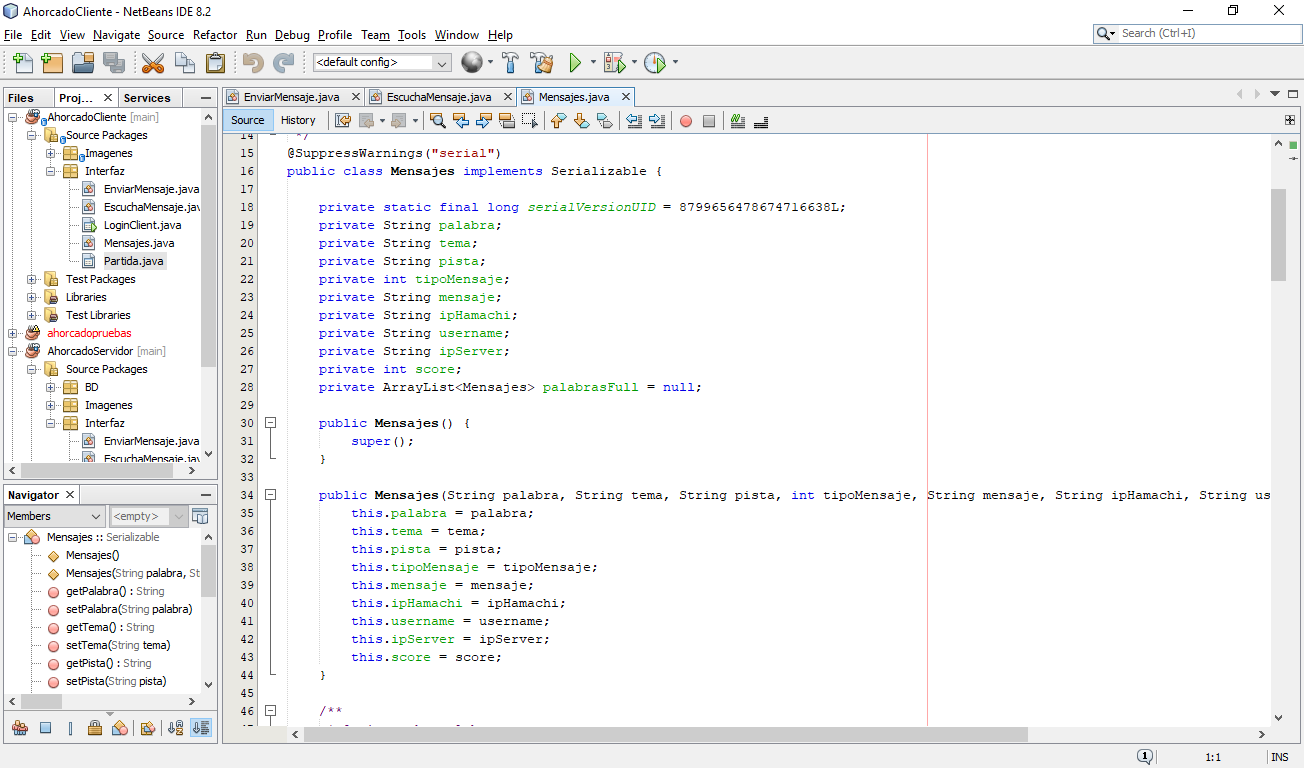


. Pantalla partida

# **Desarrollo**

Clase Mensajes

Esta clase es utilizada tanto por el cliente como el servidor. La clase tiene como atributos toda la información de quien la ejecuta. La variable tipo mensaje es para saber qué información le están enviando. La variable estática final serialVersionUID ayudara a corregir los problemas de compatibilidad de versión con el objeto que tiene el servidor, cuando el cliente envíe este objeto.

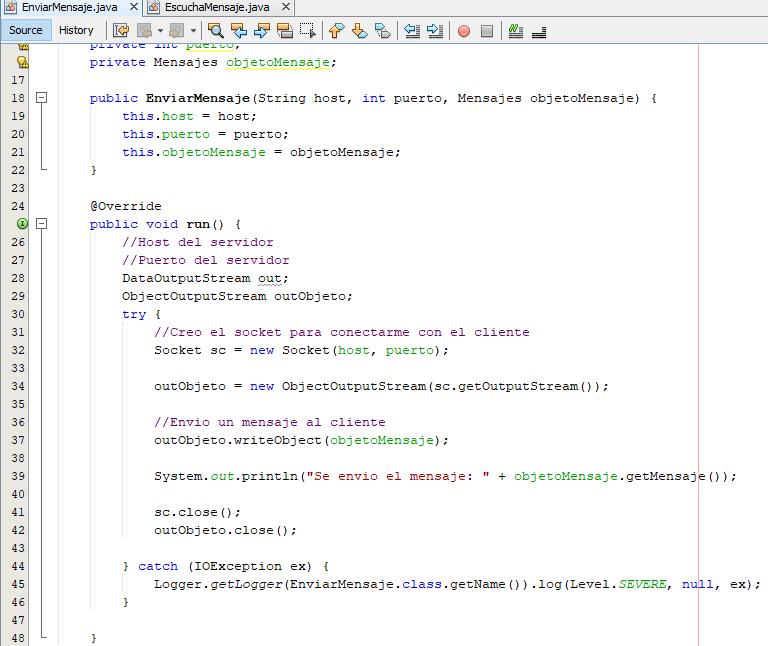


13. Desarrollo A1

EnviarMensaje

Clase que envía el mensaje al destinatario seleccionado (servidor a cliente y viceversa). Se utiliza el socket para conectarse al servidor, con su IP y el puerto. Para enviar el objeto se utiliza ObjectOutputStream para enviar este objeto (línea 34).

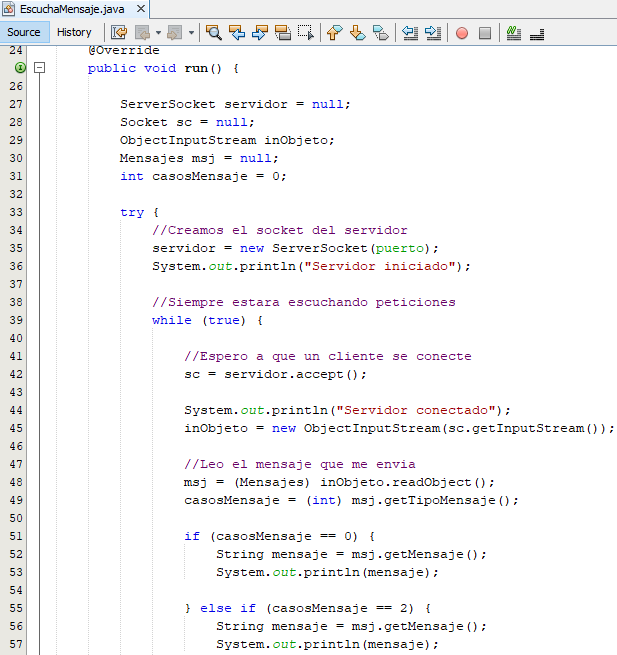
El socket para enviar el objeto se inicia y se instancia en ObjectOutputStream para enviar el objeto (línea 32 a 34). Después el objeto que se enviara se escribe en el socket (línea 37) y se envía. Finalmente se cierra la conexión del socket y del objectoutput (línea 41 y 42).



. Desarrollo A2

EscuchaMensaje

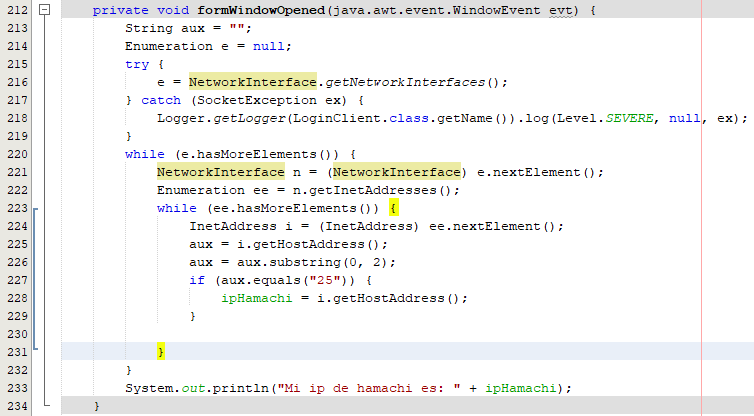
En esta clase reciben mensajes del servidor o cliente. Se utiliza la clase socket y serverSocket para recibir estos mensajes del servidor (línea 42). El objeto que se recibe del servidor tiene un casomensaje (línea 51) que sirve para identificar qué es lo que está se le está enviando. Con ObjectInputStream se recibe el objeto y este se castea a la clase local, del mismo tipo, de quien recibe (línea 45 a 50).



. Desarrollo B1

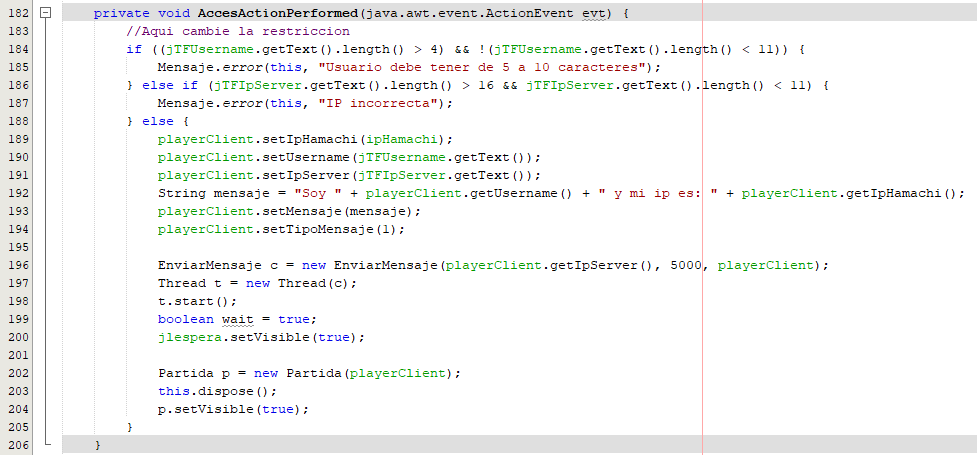
Pantalla LoginCliente

Código de la pantalla principal del jugador (cliente). Cuando se abre la ventana, se obtiene la dirección IP del jugador (línea 224) y se guarda (línea 228) para posteriormente enviar mensajes al servidor.



. Desarrollo C1

Se incluyen validaciones de dos campos, nombre de usuario y dirección IP de servidor. Si el nombre no cumple con las especificaciones o esta repetido, muestra un mensaje de error al cliente (línea 185). Cuando ingresa la IP y se intenta conectar, si el servidor no existe, muestra un error al cliente (línea 187).

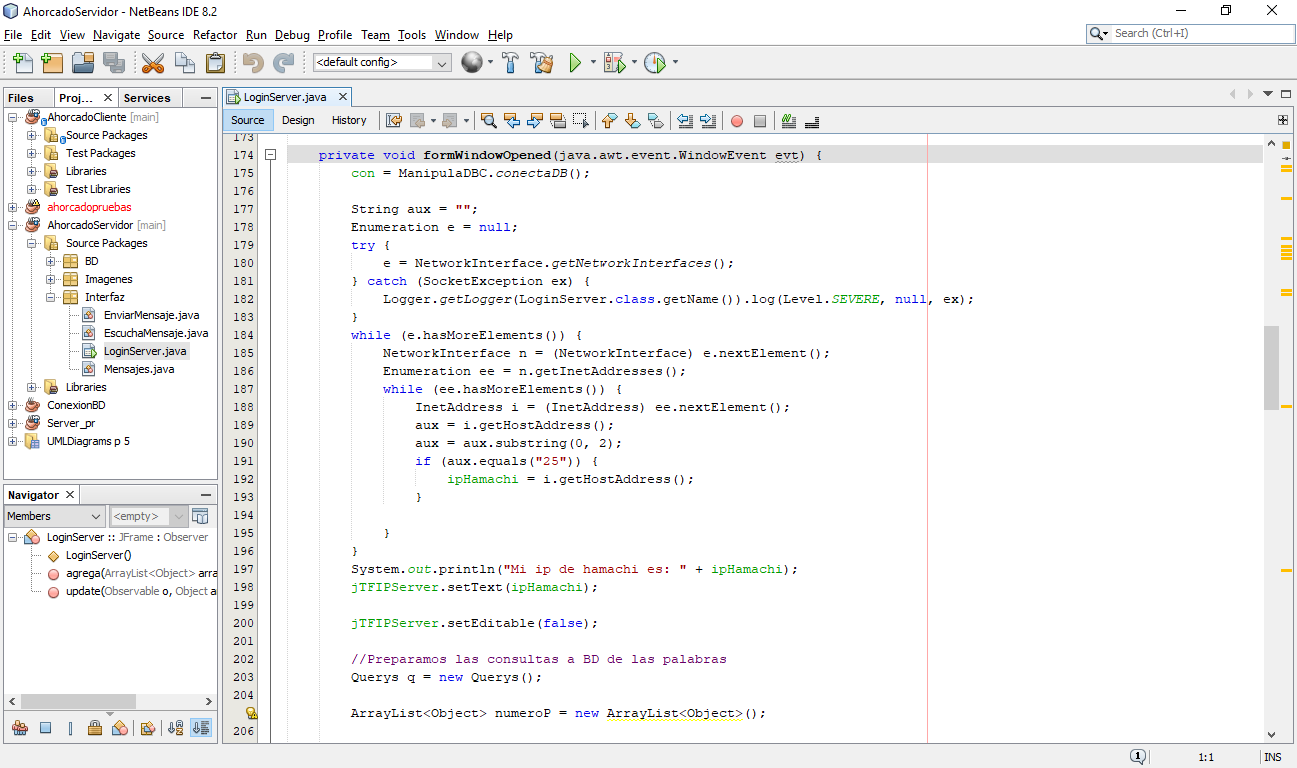


**17**. Desarrollo C2

Si la conexión es posible, se enviará el objeto EnviarMensaje al servidor, con su dirección, el username, y un mensaje (línea 106-198). Después de enviar el objeto, se abre la ventana de partida en modo espera.

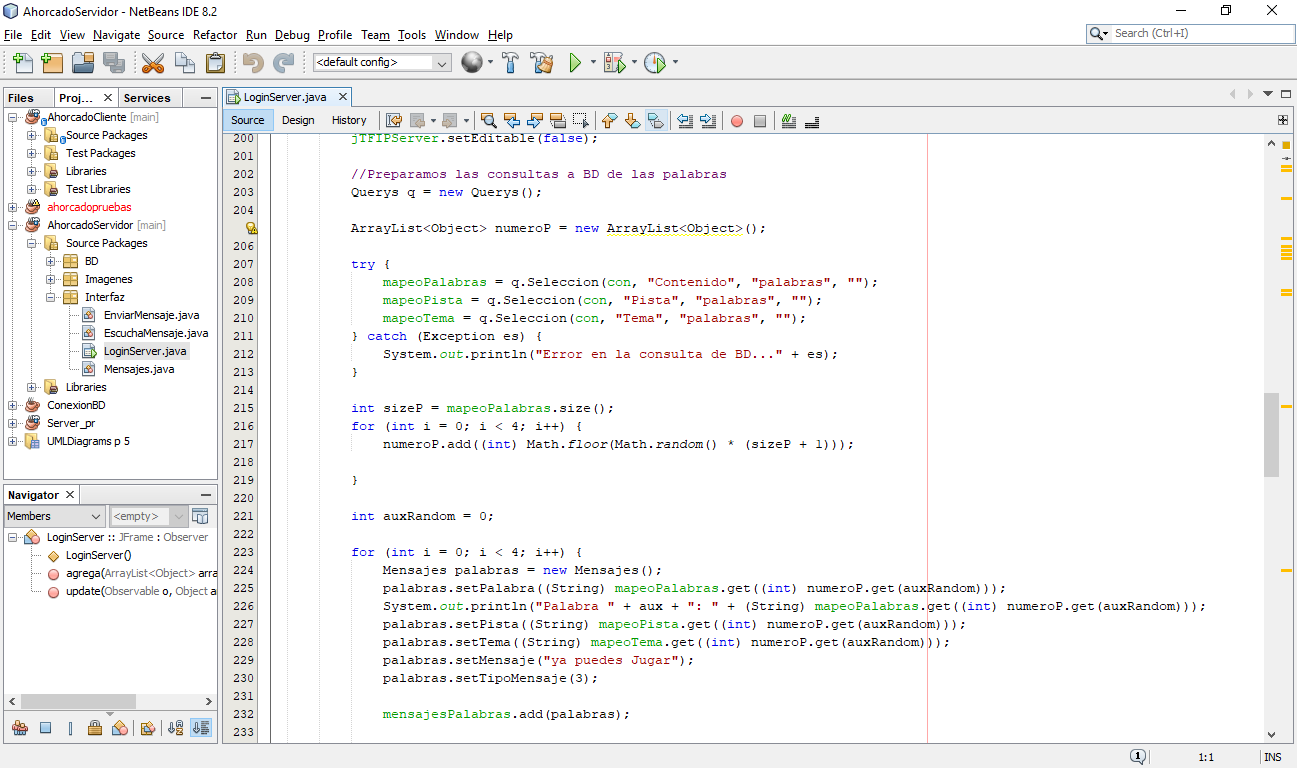
AhorcadoServidor

Cuando el servidor se enciende, se obtiene la dirección del host (línea 189).



18. Desarrollo D1

A partir de la línea 203 se preparan los querys para hacer la consulta de las palabras de la base de datos que enviara a cada jugador. Se generan números aleatorios (línea 217) 4 veces, que serán los ID´S de las 4 palabras que enviara a cada jugador.

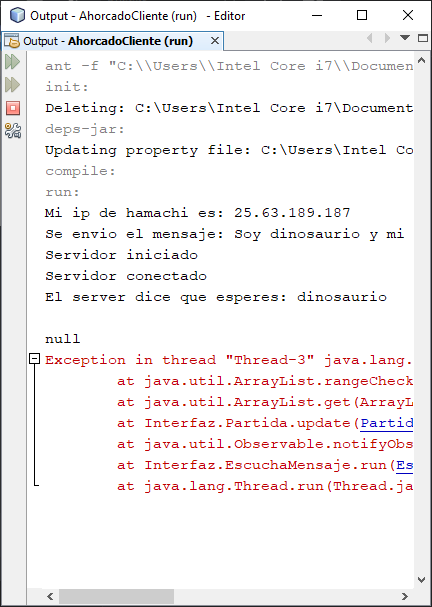


19. Desarrollo D2

Las palabras se almacenan en el objeto Mensajes y enviara el mensaje “ya puedes jugar” junto con las palabras, al cliente (línea 229 a 232).

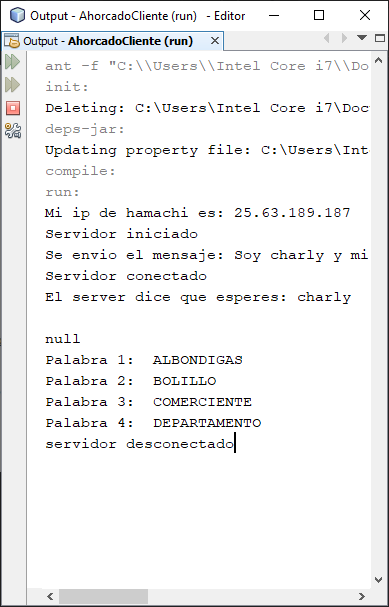
# **Experimentación**

* Prueba para recibir mensaje del servidor, avisándole al cliente que espere al segundo jugador. Problema al enviar el mensaje.



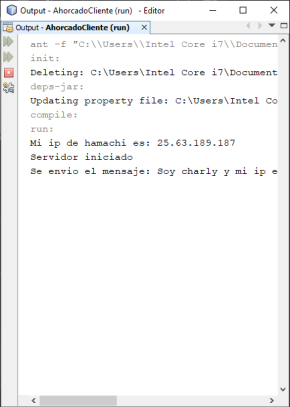
**20**. Error recibir mensaje 1

* El servidor envía el mensaje al cliente diciéndole que espere y enviándole sus palabras que tendrá que adivinar.



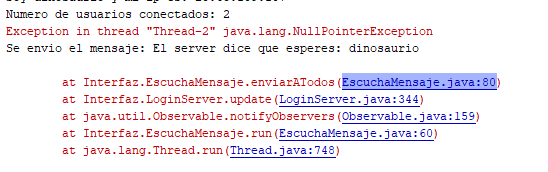
**21**. Corrección. Mensaje recibido

* El cliente no pudo recibir el mensaje del servidor, se mantiene esperando.



**22**. Error esperando mensaje

* Error en el servidor al enviar el mensaje.



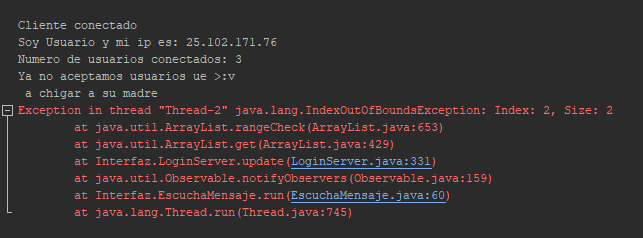
**23**. Error servidor envía mensaje

* El servidor esta aceptando las conexiones de los clientes y aumenta su contador de usuarios.



**24**. Servidor acepta clientes

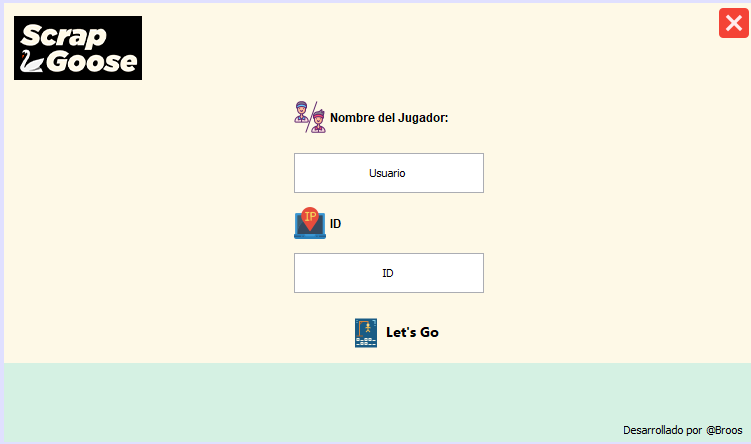
* El servidor no rechaza la conexión de un cliente cundo ya están conectados el límite de clientes.



**25**. Error conexión inesperada

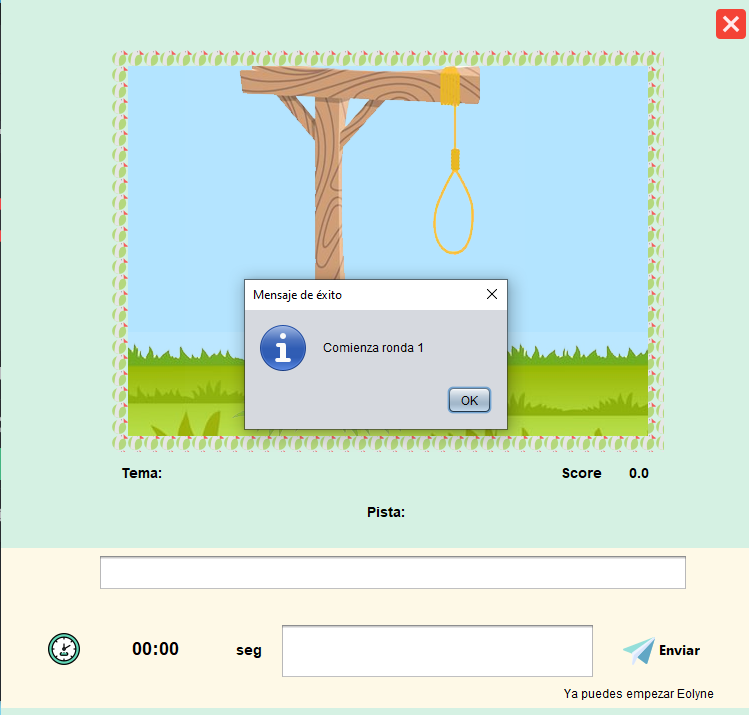
# **Resultados**

Pantalla de inicio del cliente, se ingresa el usuario y el ID(IP)

****

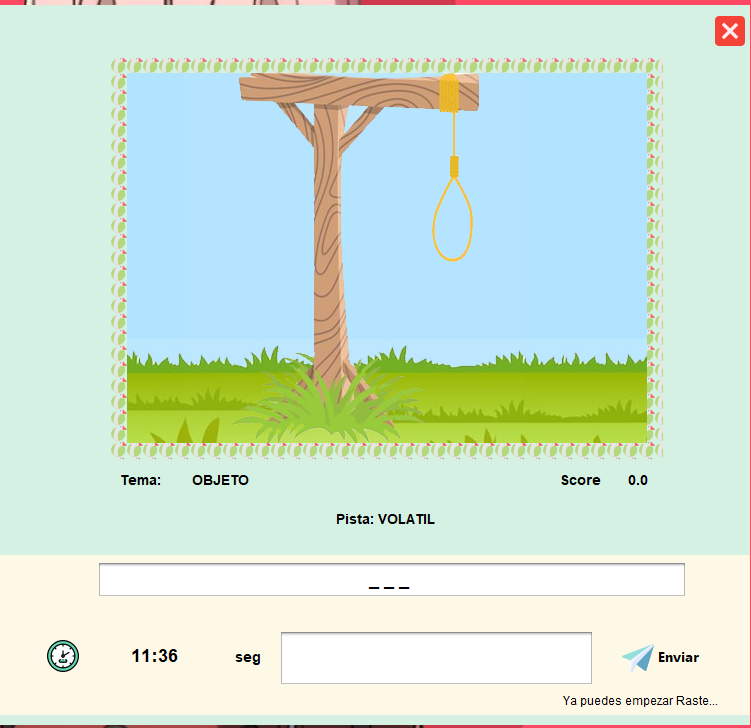
**26**. Pantalla principal

Pantalla de partida, te notifica el inicio de la ronda



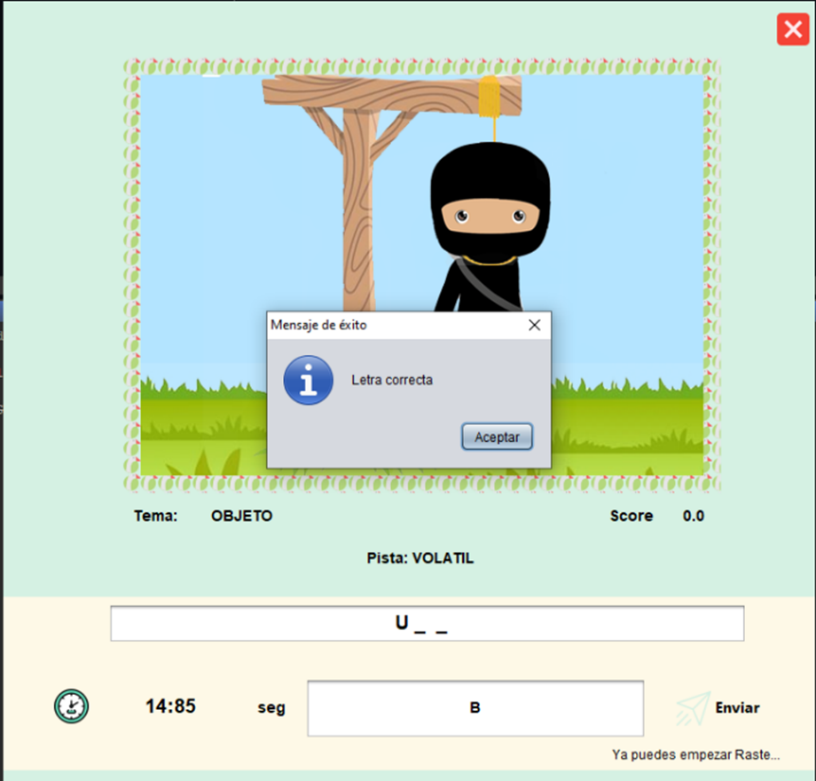
**27**. Mensaje de partida iniciada

Pantalla de partida en curso



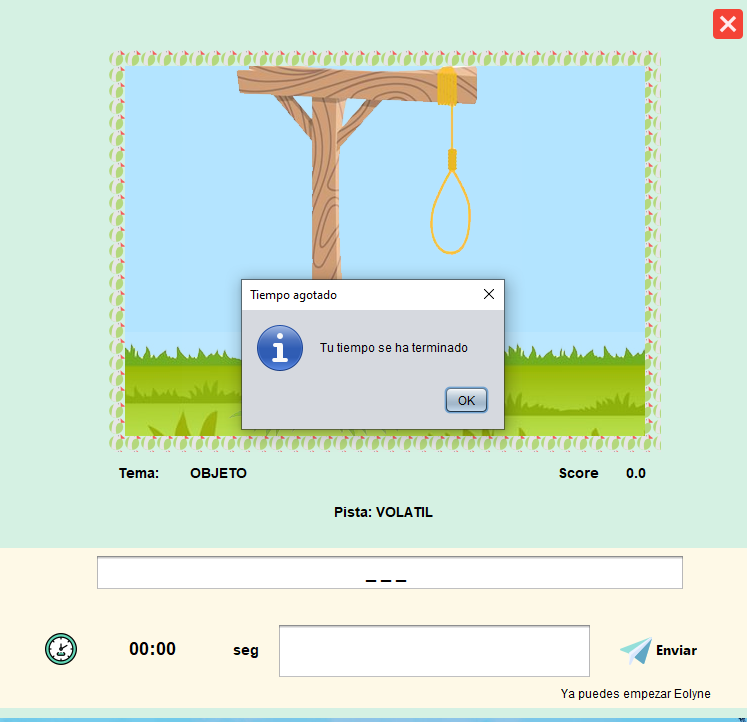
**28**. Pantalla de partida en curso

Mensaje que te notifica cuando has acertdo la letra correcta



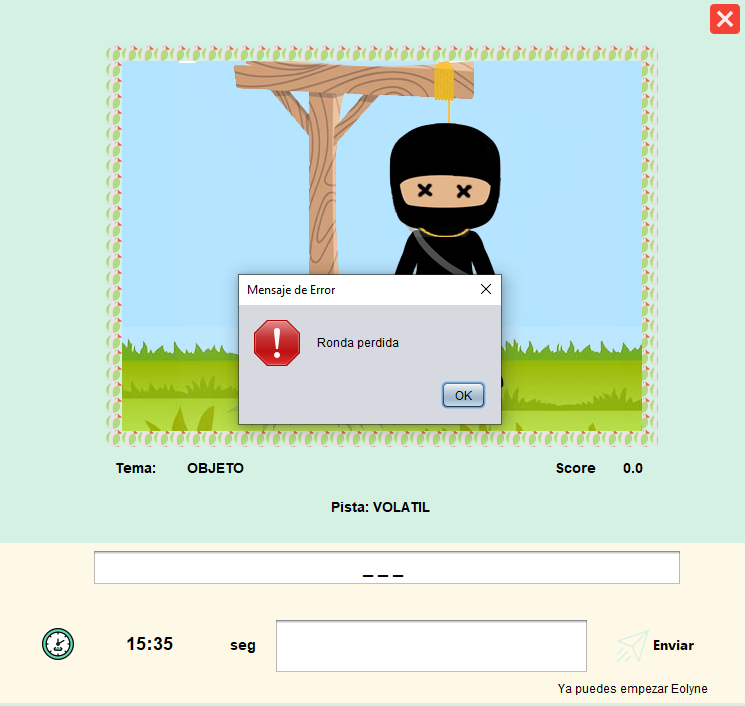
**29**. Mensaje de letra correcta

Pantalla de partida, mensaje que te notifica que se te ha acabado el tiempo



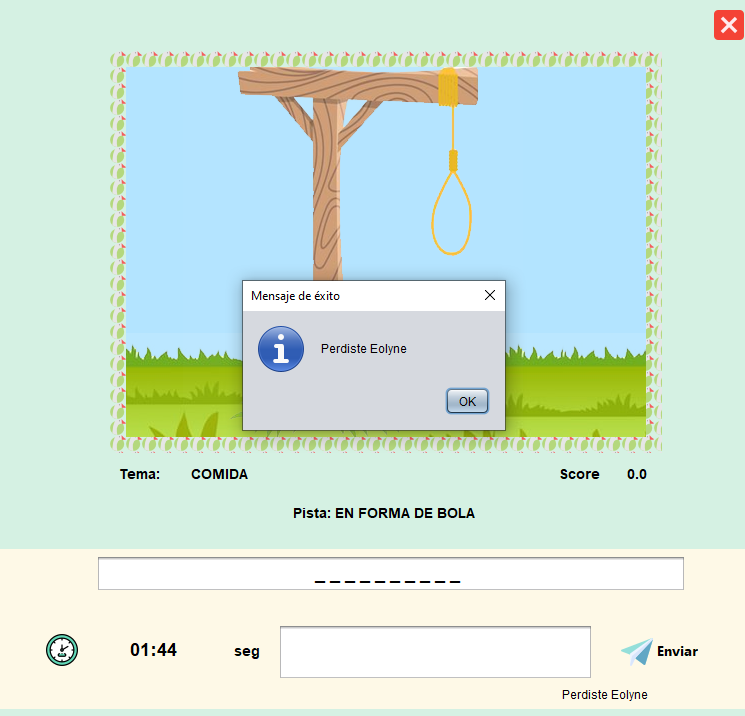
**30**. Mensaje tiempo terminado

Pantalla de partida, mensaje que te notifica que has perdido la ronda



**30**. Mensaje ronda perdida

Pantalla de partida, mensaje que te notfica que has perdido la partida



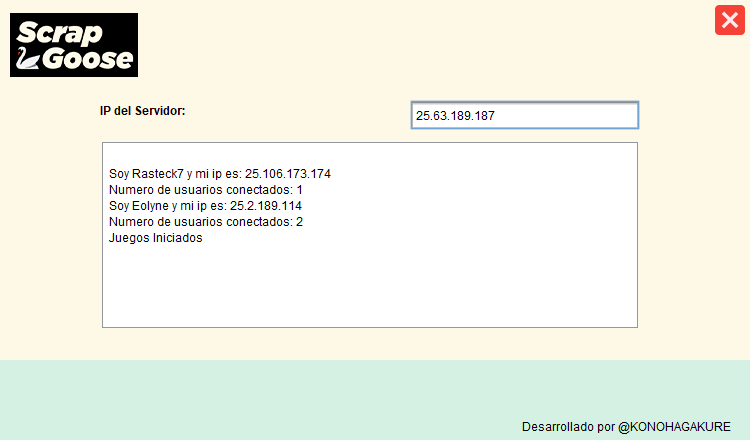
**31**. Mensaje de que perdiste

Pantalla de partida, mensaje que te notifica que has ganado la partida



**32**. Mensaje de que ganaste

Pantalla de servidor que te notifica sobre todas las conexiones existentes de usuarios



**33**. Pantalla de servidor 1



**34**. Pantalla de servidor 2

# **Conclusiones**

Se logró crear un sistema de cliente servidor, mediante el uso de sockets y por cuestiones de seguridad con la finalidad de hacer más factible el uso de la aplicación, se utilizó la aplicación hamachi, con esto se logró conectar más de tres sistemas de cómputo desde distintos puntos mediante una conexión a internet, con lo cual se logró obtener los resultados esperados, logrando la interconexión de dos usuarios los cuales pudieron interactuar y jugar el juego “ahorcado”.

A pesar de que se tuvieron varias dificultades con él envió y recibimiento de información, finalmente se logró la transmisión de datos de forma correcta.

El desarrollo del proyecto nos logró dar un gran panorama sobre el funcionamiento de un sistema distribuido, mediante el uso de sockets, así mismo se logró comprender de forma correcta sus ventajas y desventajas, y la utilidad que tienen este tipo de sistemas en diversas aplicaciones.

**Referencias**

Gomme, A. B. (2020). *The Traditional Games of England, Scotland, and Ireland: Volume 2*. BoD–Books on Demand.

Bharambe, A. R., Pang, J., & Seshan, S. (2006, May). Colyseus: A Distributed Architecture for Online Multiplayer Games. In *NSDI* (Vol. 6, pp. 12-12).

Belli, S., & Raventós, C. L. (2008). Breve historia de los videojuegos. *Athenea Digital. Revista de pensamiento e investigación social*, (14), 159-179.

Marini, E. (Octubre, 2012) *El Modelo Cliente/Servidor*. Obtenido de: https://www.linuxito.com/docs/el-modelo-cliente-servidor.pdf