

Universidad Autónoma del Estado de México

Unidad Académica Profesional Tianguistenco

Ingeniería en Software

Unidad de Aprendizaje:

Sistemas Distribuidos

Profesor:

Jovani Armaga García

Titulo:

Proyecto: Juego de Ahorcado

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

David Velázquez Ramírez

Uriel García Martínez

Edwin Misael Vázquez Rivera

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

Contenido

[Introducción 4](#_Toc55815539)

[Antecedentes 4](#_Toc55815540)

[Historia del juego del ahorcado 4](#_Toc55815541)

[Videojuegos y sistemas distribuidos 4](#_Toc55815542)

[Planteamiento del problema 5](#_Toc55815543)

[Objetivos generales 5](#_Toc55815544)

[Objetivos específicos 6](#_Toc55815545)

[Hipótesis 6](#_Toc55815546)

[Justificación 7](#_Toc55815547)

[Delimitación del problema 7](#_Toc55815548)

[Definición de requerimientos 7](#_Toc55815549)

[Requerimientos Funcionales 8](#_Toc55815550)

[Requerimientos No Funcionales 9](#_Toc55815551)

[Herramientas por utilizar 14](#_Toc55815552)

[Arquitectura SD 15](#_Toc55815553)

[Diseño 16](#_Toc55815554)

[Prototipo 16](#_Toc55815555)

[Diagrama de casos de uso 17](#_Toc55815556)

[Diagrama de secuencias 18](#_Toc55815557)

[Diagrama de bloques 19](#_Toc55815558)

[Diagrama entidad-relación 20](#_Toc55815559)

[Diccionario de datos 21](#_Toc55815560)

[Tablas relacionales 21](#_Toc55815561)

[Descripción de tablas 22](#_Toc55815562)

[Interfaces 23](#_Toc55815563)

[Desarrollo 24](#_Toc55815564)

[Experimentación 25](#_Toc55815565)

[Resultados 26](#_Toc55815566)

[Conclusiones 27](#_Toc55815567)

**Índice de Ilustraciones**

[1Diseño. Prototipo pantalla principal 15](#_Toc55815605)

[2Diseño. Prototipo pantalla de partida 15](#_Toc55815606)

[3. Diagrama Entidad-Relación 19](#_Toc55815607)

[4 Tablas relacionales 20](#_Toc55815608)

# **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**

# **Delimitación del problema**

# **Definición de requerimientos**

# **Requerimientos Funcionales**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | RF1 | | |
| **Nombre** | Ingresar usuario | | |
| **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: 15 * 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.   Botón entrar: al presionar, muestra la pantalla de sala.  Casos:  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”. | | |
| **Prioridad de Requerimiento** | Alta | **Exigencia de requerimiento** | Exigible |
| **Relación** | RF2 | | |
| **Editor** | Carlos Jonathan López Palma | | |
| **Fecha** | 21 de octubre de 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 | | |

### Herramientas por utilizar

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:

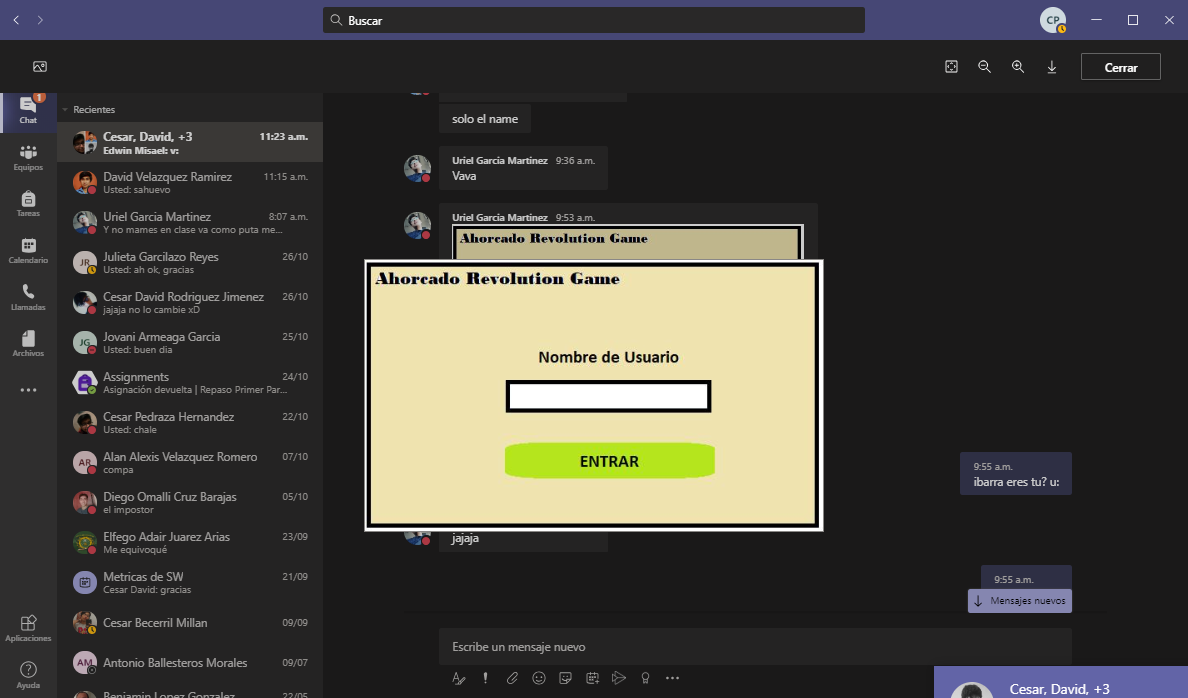
* Socket: Implementa un extremo de la conexión TCP.
* ServerSocket: Se encarga de implementar el extremo Servidor de la conexión en la que se esperarán las conexiones de los clientes.
* DatagramSocket: Implementa tanto el servidor como el cliente cuando se utiliza UDP.
* DatagramPacket: Implementa un 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 generacion de interfaces y requiere de una Instalación y actualización simple.

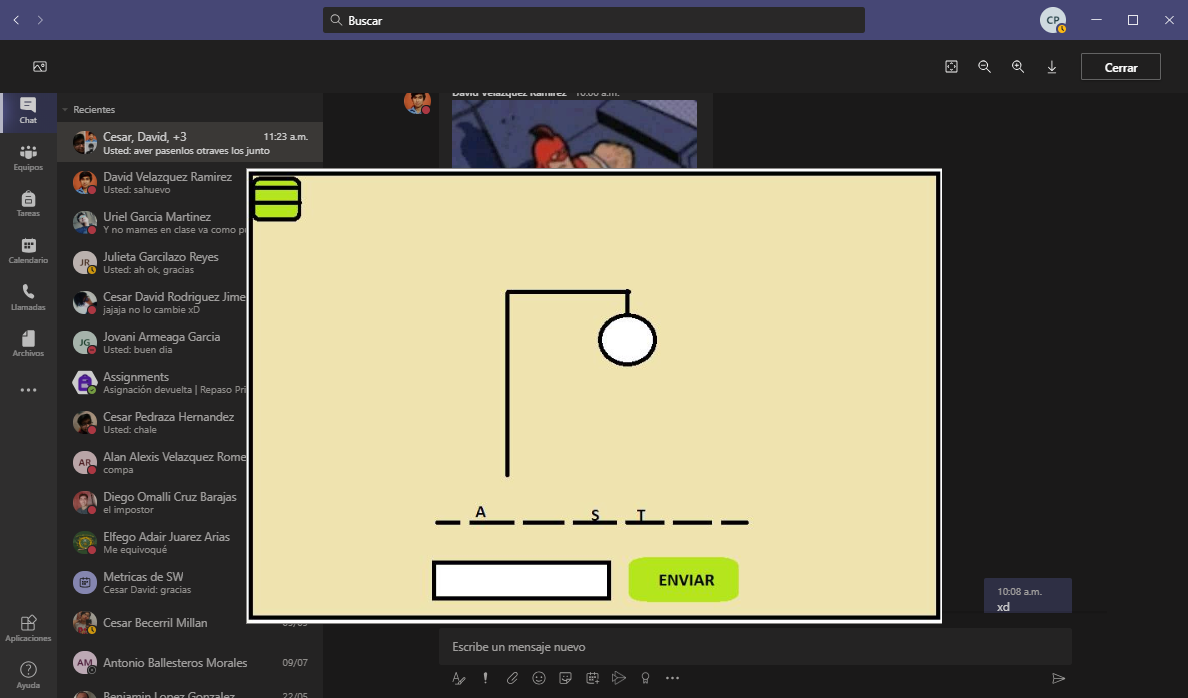
# **Arquitectura SD**

# **Diseño**

## Prototipo



1Diseño. Prototipo pantalla principal



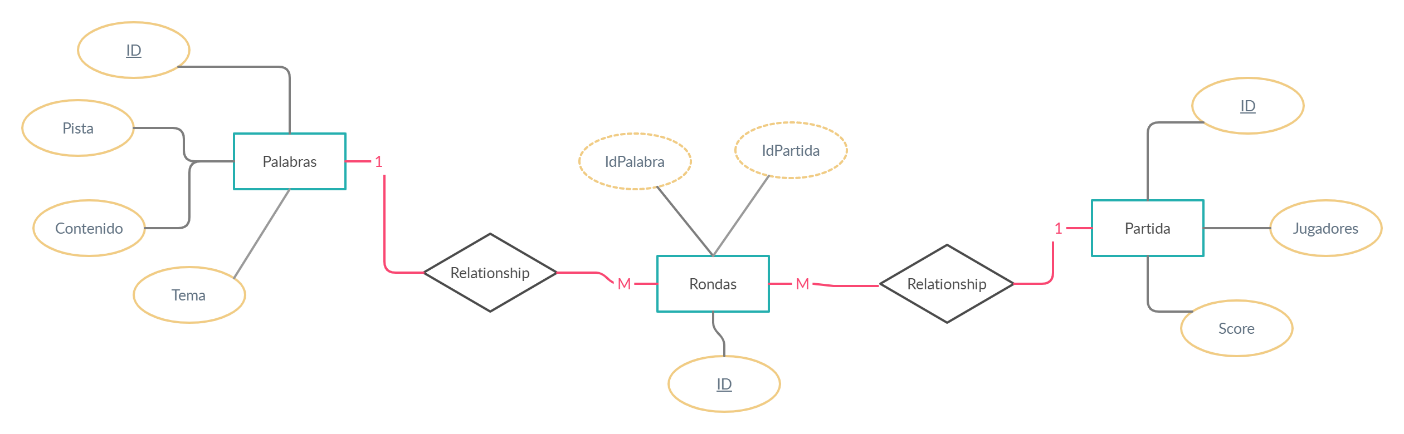
2Diseño. Prototipo pantalla de partida

# **Diagrama de casos de uso**

# **Diagrama de secuencias**

# **Diagrama de bloques**

# **Diagrama entidad-relación**



. Diagrama Entidad-Relación

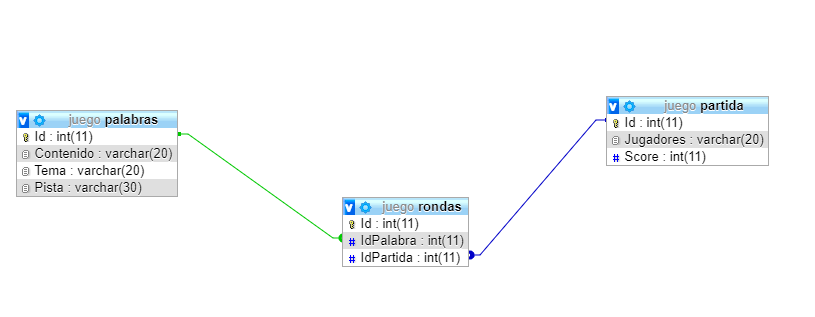
# **Diccionario de datos**

|  |  |
| --- | --- |
| **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’ |

|  |  |
| --- | --- |
| **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 |

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

# **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’ |

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 |

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 |

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**

# **Desarrollo**

# **Experimentación**

# **Resultados**

# **Conclusiones**

**Referencias**