REPORTE DE ESTADO

## Trabajo Práctico “Laberintos y Fantasmas”

#### Grupo Pixelados

### Cátedra:

Algoritmos y Estructuras de Datos

### Integrantes:

* Ermasi, Franco
* Ibarra Sánchez, Ludmila Daiana
* Maranzana, Rodrigo Ezequiel
* Masino, Carlos Nicolas

Fecha: 24/09/2025

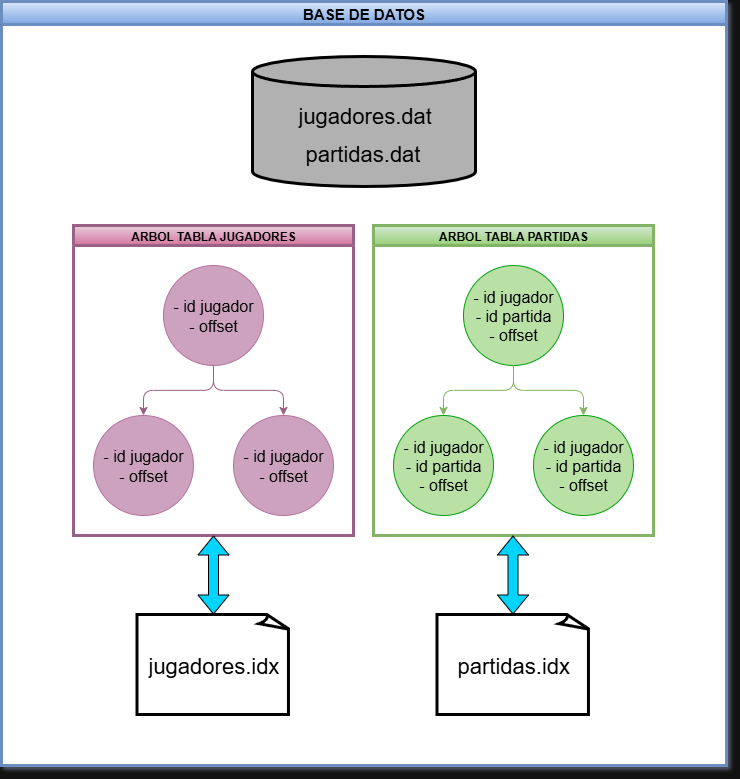
## Funcionalidades actuales del Cliente:

Nuestro grupo tomó la decisión de utilizar la biblioteca gráfica SDL y sus derivadas SDL\_mixer (mezclador de audio), SDL\_ttf (soporte TrueType) y SDL\_image (decodificador de imagenes).

Las siguientes funcionalidades se encuentran completas y funcionando:

* Lectura de configuraciones iniciales desde un archivo de texto.
* Generación del laberinto utilizando el algoritmo DFS (Deep-First Search) y post procesado, para asegurar la jugabilidad y controlar la dificultad.
* Colocación de puerta de entrada y salida, asegurando su conexión con el laberinto.
* Colocación de premios y vidas extra, puestas en lugares estratégicos y su recolección por parte del jugador, junto a su otorgamiento respectivo de punto/vida.
* Utilización de semillas que permiten generar partidas y sus sucesivas rondas exactamente iguales.
* Colocación de hasta la cantidad de fantasmas indicada en el archivo de configuración, variando ronda a ronda entre 1 fantasma y el máximo establecido.
* Cuatro tipos de fantasmas con sus respectivas variaciones de búsqueda del jugador, utilizando intercalación entre el algoritmo DFS (Deep-First Search) y BFS (Breadth-First Search), siendo el último mucho más agresivo aumentando así drásticamente la dificultad del juego. Cada fantasma posee su propio color para su fácil detección.
* Encolado de los movimientos del jugador y los fantasmas, con su posterior función de procesado de movimientos.
* Registro de los movimientos del jugador realizados en cada ronda y su posterior muestra por consola.
* Menú principal con las opciones de Nueva Partida, Continuar, Cambiar Usuario, Estadísticas y Salir (Cambiar Usuario y Estadísticas aún en progreso)
* Ingreso de nombre de Usuario, al iniciar el juego, mediante una ventana con un campo de texto realizado con SDL.
* Servicio de conexión socket tanto en cliente como el servidor.
* Servidor con una base de datos conformada por dos tablas, jugadores y partidas, guardadas cada una en su respectivo archivo .dat e implementada la búsqueda de los registros mediante dos árboles de búsqueda binaria los cuales persisten en sus respectivos índices. A su vez se cuenta con una lista de comandos para la comunicación del cliente al servidor. (Trabajo en Progreso)

## Diseño de la Base de Datos del Servidor



## Funcionalidades actuales del Cliente:

El diseño de la base de datos que utilizará creará el cliente en el servicio de Base de Datos está compuesto por dos tablas almacenadas en dos archivos binarios. Cada uno cuenta con su respectivo índice implementado con árbol de búsqueda binaria, los cuales persisten en dos archivos idx.

La comunicación entre cliente-servidor se lleva a cabo mediante comandos de texto. El servicio de base de datos tiene comandos (CREAR, INSERTAR, ABRIR, ACTUALIZAR, SELECCIONAR), palabras clave (TABLA, EN, DONDE, TODO), operadores (IGUAL, MAYOR, MENOR, DISTINTO), restricciones (PK) y tipos de datos (ENTERO, FLOTANTE, TEXTO).

La siguiente lista incluye ejemplos de las solicitudes admitidas por el servidor:

* CREAR:
  + CREAR TABLA jugadores (idJugador ENTERO, nombre TEXTO(16), puntajeMax ENTERO, PK(idJugador))
  + CREAR TABLA partidas (idPartida ENTERO, puntaje ENTERO, PK(idJugador, idPartida))
* INSERTAR:
  + INSERTAR EN jugadores (idJugador 21, nombre PEPE, puntajeMax 15)
* ABRIR:
  + ABRIR TABLA jugadores
* ACTUALIZAR:
  + ACTUALIZAR EN jugadores (puntajeMax 25) DONDE idJugador IGUAL 21
* SELECCIONAR:
  + SELECCIONAR TODO jugadores DONDE idJugador IGUAL 21

## Bibliotecas del Cliente:

* **JUEGO**: Controla la creación e inicialización de los componentes de SDL necesarios. Gestiona el game loop y los respectivos llamados a las funciones que conforman tanto a la lógica, detección de inputs, menús, comunicación con el socket, reproducción de efectos de audio y llamado a renderizado.
* **LOGICA**: Controla, como su nombre indica, la lógica que Laberintos y Fantasmas debe seguir. Gestiona los llamados a funciones generadoras de la partida, los movimientos del jugador, de los fantasmas, el otorgamiento de vidas, puntos, mientras regla las interacciones.
* **ASSETS**: Carga desde el disco los archivos de imagen, audio y fuentes TrueType necesarios para la ejecución del juego.
* **DIBUJADO**: Se encarga del Algoritmo del Pintor para la correcta muestra del mundo y las entidades que la componen. Como el juego utiliza perspectiva Top Down, se requiere un dibujado desde la fila menor, columna a columna, hasta la fila mayor, intercalando el dibujado de las entidades para la generación del efecto de 2.5D.
* **ESCENARIO**: Gestiona los algoritmos de generación del laberinto y la inicialización del escenario de juego.
* **CLIENTE**: Gestiona la comunicación con el servidor para la inserción y recupero de los jugadores y sus respectivas partidas utilizando la biblioteca winsock2.
* **VENTANA**: Permite la generación de ventanas graficas de SDL genéricas dentro de la propia ventana del juego.
* **WIDGET**: Permite la creación de utilidades como campos de texto, contadores numéricos gráficos, gráficos de barra, etc. para la muestra y/o recolección de información durante el juego y la partida.
* **MENU**: Permite la creación de menús gráficos interactivos y personalizables.
* **TEXTO**: Gestiona la creación de texturas a partir de strings y fuentes TrueType.
* **GRAFICOS**: Gestiona la escritura de texturas sobre el framebuffer de SDL.
* **TEMPORIZADOR**: Gestiona temporizadores (no son multihilo) que permiten ejecutar acciones de poca precisión como la animación de las entidades y entornos y la espera no bloqueante.
* **ARCHIVO**: Funciones para la lectura, parseo y escritura del archivo de configuración.
* **PILA, COLA, ARBOL**: Funciones de sus respectivos TDA.

# Diagrama de Bloques actual

