UNIVERSIDAD AUSTRAL DE CHILE
INSTITUTO DE INFORMATICA

Informe Arquitectura de software
Grupo 3 Buscaminas

Nombres: Nicolas Donoso , Cristian Huenchullanca , Ivan Pizarro
Fecha de entrega : 18 de diciembre de 2023

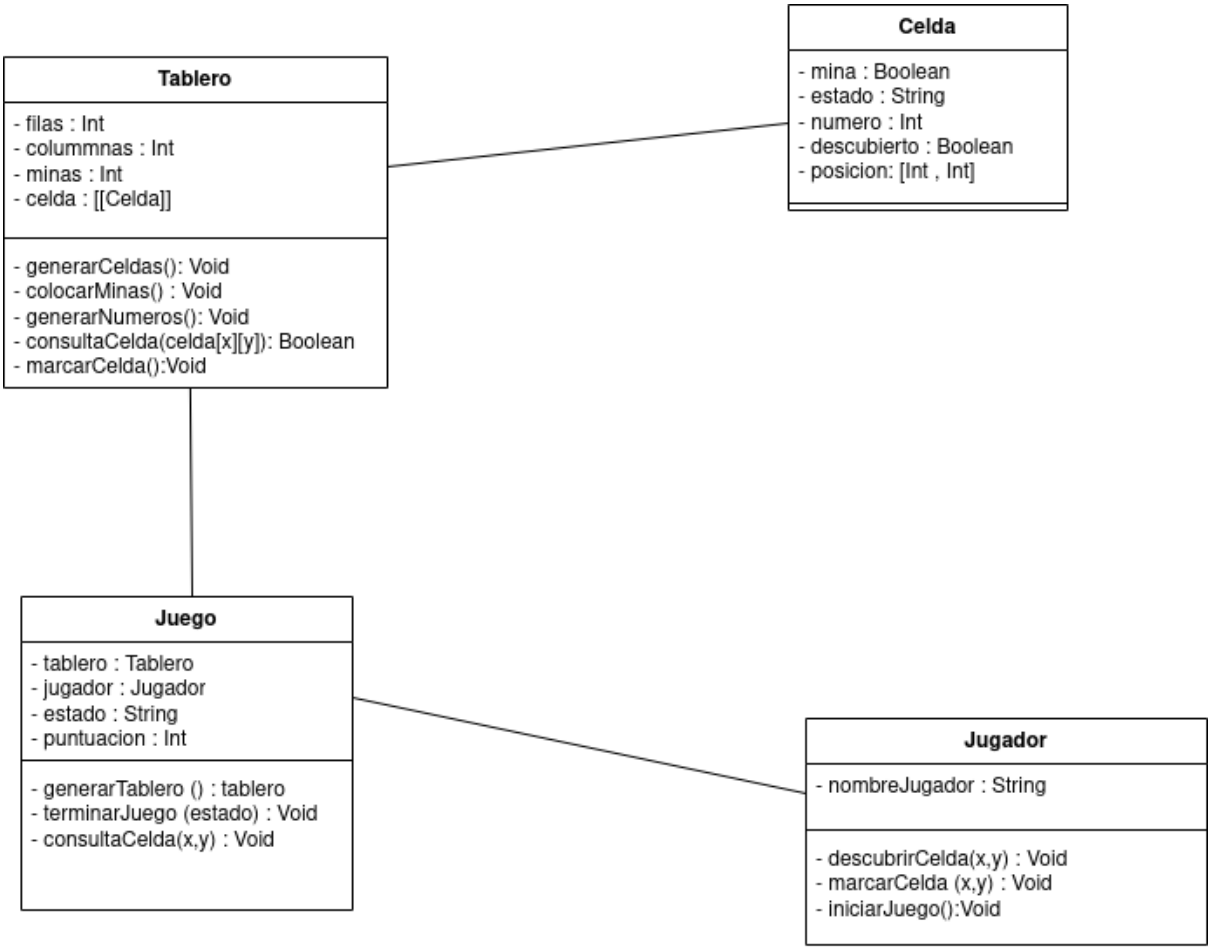
UNIVERSIDAD AUSTRAL DE CHILE
INSTITUTO DE INFORMATICA

Introduccion

En el presente informe, se abordará el desarrollo de un proyecto centrado en el juego Buscaminas. Para llevar a cabo este proyecto, se utilizará la metodología 4+1, un enfoque de investigación que combina cuatro fases fundamentales: planificación, ejecución, análisis y comunicación, con una fase adicional de evaluación.

Este informe pretende no solo describir el proceso de desarrollo del proyecto, sino también reflexionar sobre las ventajas y desafíos de la metodología 4+1. Esperamos que este trabajo contribuya a la comprensión y la apreciación del juego Buscaminas, así como a la mejora de las prácticas de investigación en el campo de los videojuegos.

METODOLOGIA 4+1
Diagrama de clase :



Tenemos 4 tablas para nuestro diagrama de clase en la cual tenemos primeramente :

Clase Tablero:

- Atributos:
    - **filas:** Un entero que indica el número de filas en el tablero.
    - **columnas:** Un entero que indica el número de columnas en el tablero.
    - **minas:** Un entero que indica la cantidad total de minas en el tablero.
    - **celda:** Una matriz bidimensional del tipo Celda, representando cada celda individual en el tablero.
- Métodos:
    - **generarCeldas():** Inicializa las celdas del tablero.
    - **colocarMinas():** Coloca las minas aleatoriamente en las celdas del tablero.
    - **generarNumeros():** Asigna números a las celdas basados en la cantidad de minas adyacentes.
    - **consultaCelda(celda[x][y]):** Retorna un booleano indicando si la celda consultada contiene una mina o no.
    - **marcarCelda():** Marca una celda específica.

Clase Celda:

- Atributos:
    - **mina:** Un booleano que indica si la celda contiene una mina (true) o no (false).
    - **estado:** Una cadena que representa el estado actual de la celda (por ejemplo, descubierta, oculta, marcada).
    - **numero:** Un entero que indica cuántos vecinos contienen mina
    - **descubierto:** Booleano para saber si ya fue revelado lo que hay en esa casilla
    - **posicion:** Array con dos enteros indicando su posición exacta dentro del array bidimensional del Tablero

Clase Jugador:

- Atributos:
    - **nombreJugador :** String para almacenar nombre del jugador
- Métodos:
    - **descubrirCelda():** Método para descubrir una celda en el tablero.
    - **marcarCelda():** Metodo para poder marcar una celda con una bandera (u otro)
    - **iniciarJuego():** Metodo que se utiliza para poder iniciar la partida .

Clase Juego:

Atributos:

- tablero : Tablero: Indica que el juego tiene un tablero asociado, probablemente donde se juega el juego.
- jugador : Jugador: Representa al jugador que está jugando el juego.
- estado : String: Podría indicar el estado actual del juego, como “en progreso”, “ganado” o “perdido”.
- puntuacion : Int: Almacena la puntuación actual o final del jugador.

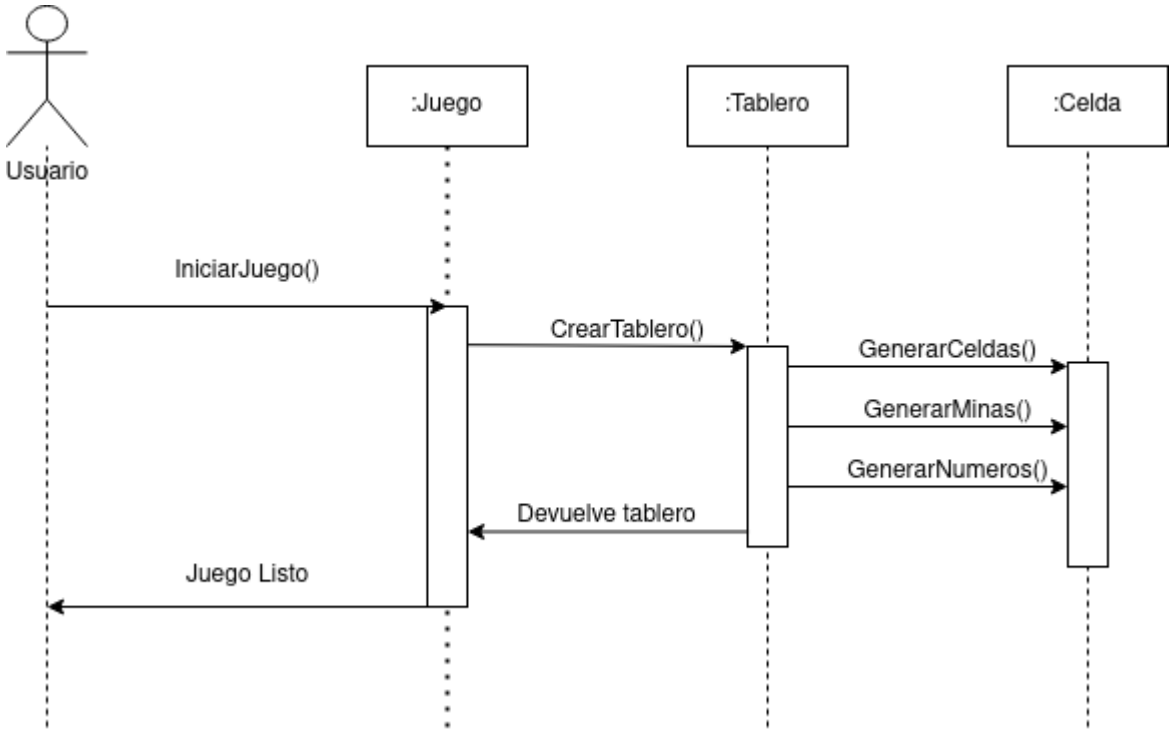
Métodos:

- generarTablero() : tablero: Un método para generar o iniciar un nuevo tablero para el juego.
- terminarJuego(estado) : Void: Este método podría ser utilizado para terminar el juego, aceptando un estado (como “ganado” o “perdido”) como parámetro.
- consultaCelda(posicion) : Void: Podría ser utilizado para consultar una celda específica en el tablero basada en su posición recibia por el jugador .

Diagrama de secuencias

Para este diagrama ocuparemos 2 en las que una consiste en la generacion del tablero y el otro es un loop en donde el jugador , estara jugando a la partida del buscaminas.

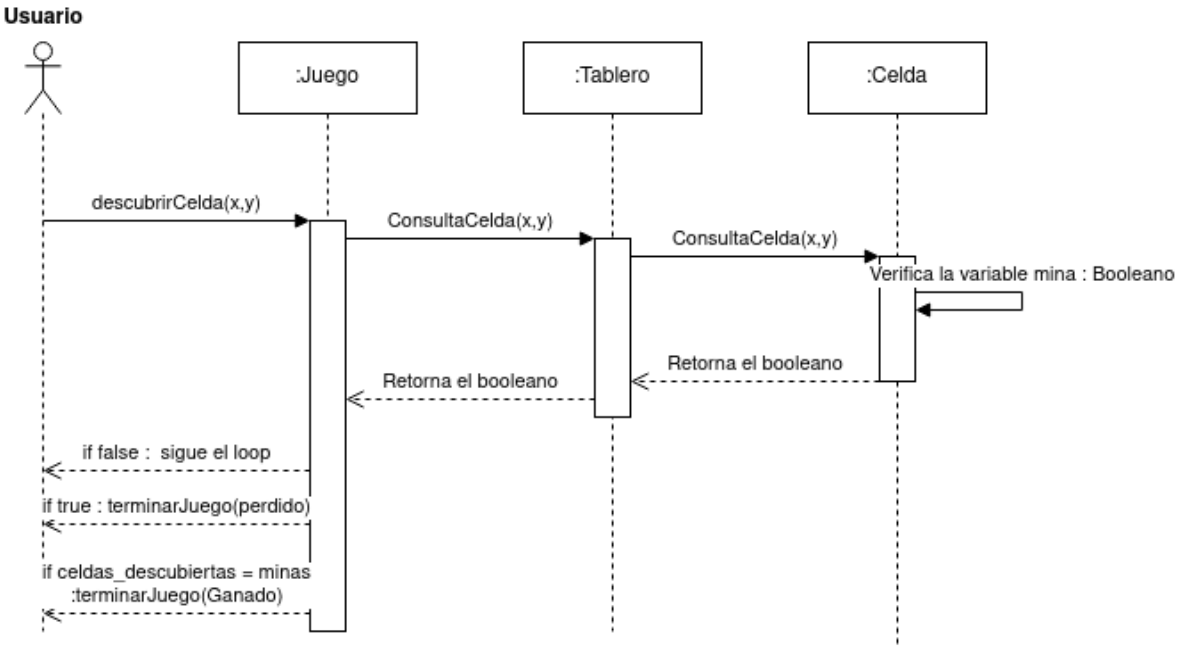
Primeramente vemos la primera diagrama de secuencias en la que se muestra la generacion del tablero , es decir cuando se inicia el juego se inician estos metodos para la generacion de este.



Primeramente el jugador , inicia la partida , por lo cual se llama a la funcion de crearTablero() , en donde este se generan las celdas , secuencialmente las minas y por último los numeros alrededor de este .

Para últimamente devolver el tablero creado , teniendo el juego listo para empezar la partida.

El segundo diagrama de secuencia se centra en cómo funciona lo que es cuando se descubre una celda , más específicamente cuando el jugador presiona la casilla para descubrir si la casilla está vacía o es una mina.



Para esta funcion es necesario saber que es un loop , es decir que siempre se va estar verificando si el usuario le da para descubrirCelda() , es decir el click en las celdas. A no ser que el juego se encuentre terminado .

Cuando el usuario le da a descubrirCelda , con las coordenadas de estas , este se va consultando en cada clase , en donde finalmente llega a la celda con las coordenadas y en esta clase se consulta acerca del valor booleano de mina , que es donde se verifica que en esa casilla tenga una mina o no . Este valor booleano se retorna hasta la clase juego , y en este se verificará si este valor es true o false , Por lo que se tendrian estos casos :

- Si el valor es falso : Quiere decir que en esa celda no hay una mina por lo que se continuará el juego , obviamente con las casillas actualizadas.
- Si el valor es true : Se tiene que la celda presionada es una mina por lo que el juego terminará .
- Por último se hace una comprobacion para saber si las celdas descubiertas son igual a la cantidad de minas , por lo que si llegara a hacer true , quiere decir que el jugador descubrio todas las casillas sin minas , por lo que gano.

UNIVERSIDAD AUSTRAL DE CHILE
INSTITUTO DE INFORMATICA

DIAGRAMA DE CASO DE USOS

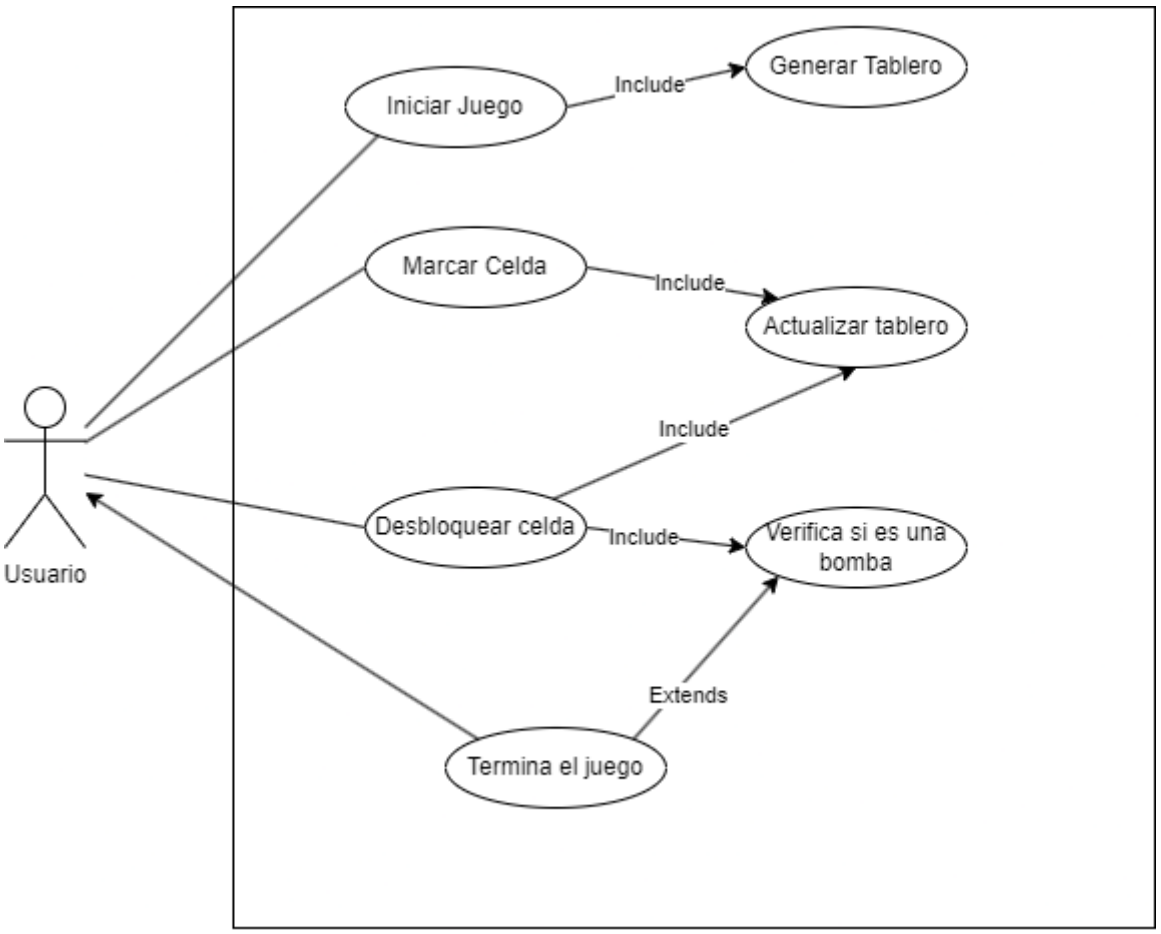


Diagrama de Componentes

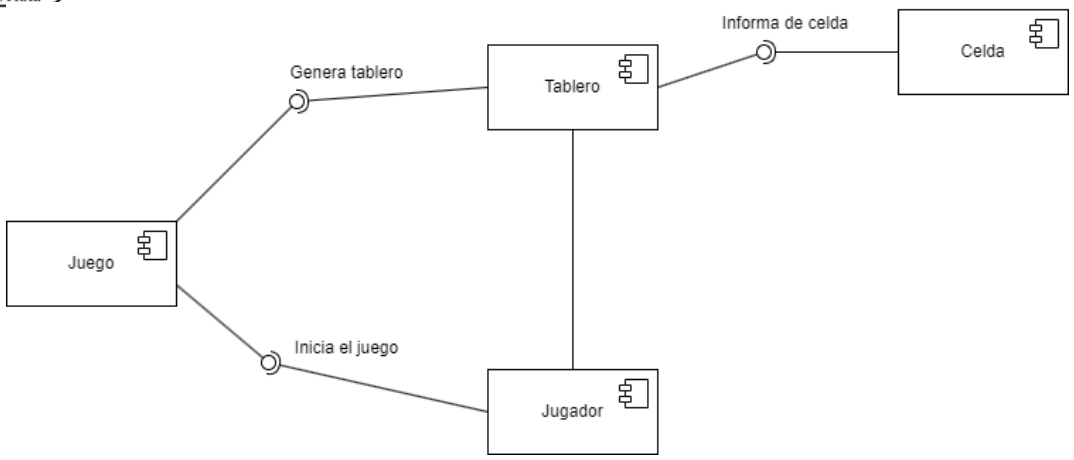
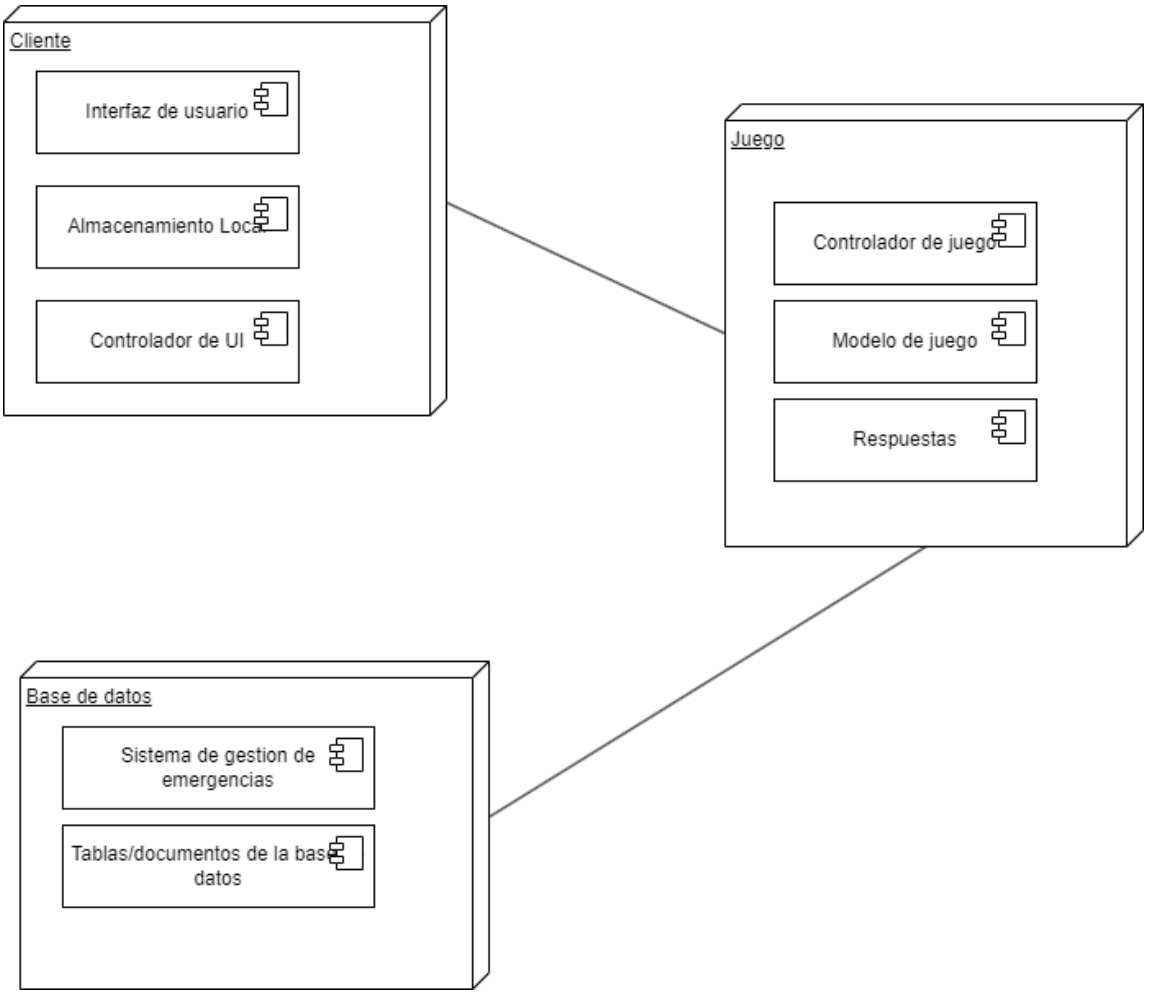


Diagrama de despliegue



El diagrama de despliegue consta de tres bloques principales:

- Cliente
- Juego

- Base de Datos.

El Cliente es el dispositivo del jugador y contiene la Interfaz de Usuario (UI), el Controlador de UI y el Almacenamiento Local.

1. Cliente: Este es el dispositivo del jugador donde se ejecuta la interfaz de usuario del juego. Contiene los siguientes elementos:
    - **Interfaz de Usuario (UI)**: Esta es la interfaz gráfica con la que el jugador interactúa. Mostraría el tablero del juego y proporciona controles para que el jugador realice acciones como descubrir una celda o marcar una celda.
    - **Controlador de UI**: Este componente maneja las interacciones del usuario con la UI. Traduciría las acciones del usuario en solicitudes que se enviarán al servidor de juego.
    - **Almacenamiento Local**: Este componente se encargaría de almacenar información localmente en el dispositivo del cliente. Por ejemplo, podría almacenar las preferencias del jugador o el estado del juego si el jugador quiere pausar y reanudar el juego más tarde.
    -

El Servidor de Juego maneja la lógica del juego y contiene el Controlador de Juego, el Modelo de Juego y Respuestas.

2. Servidor de Juego: Este es el servidor donde se ejecuta la lógica del juego. Contiene los siguientes elementos:
    - **Controlador de Juego**: Este componente maneja la lógica del juego. Procesa las solicitudes del cliente, como iniciar un nuevo juego, descubrir una celda o marcar una celda, y actualiza el estado del juego en consecuencia.
    - **Modelo de Juego**: Este componente representa el estado del juego. Contiene la información del tablero, como la ubicación de las minas y el estado de cada celda, así como la información del jugador, como la puntuación.
    - **Respuestas**: Este componente se encarga de la comunicación entre el juego y los clientes. Recibe las solicitudes del cliente y envía las respuestas.

La Base de Datos en el Servidor de Juego almacena y recupera los datos del juego.

3. Base de Datos: Esta es la base de datos donde se almacenan los datos del juego. Contiene los siguientes elementos:
    - **Sistema de Gestión de Base de Datos (DBMS)**: Este es el software que se utiliza para gestionar la base de datos. Podría ser un DBMS

relacional como MySQL o PostgreSQL, un DBMS NoSQL como MongoDB, o cualquier otro sistema que elijas.

- **Tablas/Documentos de la Base de Datos**: Estos son los lugares donde se almacenan los datos. Por ejemplo, podrías tener una tabla para los jugadores, donde cada fila representa a un jugador y tiene columnas para cosas como el nombre del jugador y la puntuación más alta. Similarmente, podrías tener otra tabla para los juegos, donde cada fila representa un juego y tiene columnas para cosas como el estado del juego y la puntuación.

Buenas prácticas

Con respecto al código y las buenas prácticas, recalcar en un principio que se adaptó en pequeñas medidas el código para que pueda implementarse en pygame. Las buenas prácticas se mencionan a continuación.

- Correcta estructura:

  El código es modular con funciones bien definidas, además cuenta con clases separadas y de fácil distinción, lo que hace que sea de fácil lectura y comprensión del funcionamiento de este.

- Documentación:

  Cuenta con comentarios incluidos en el código para una mayor compresión, ya que describen propósitos y parámetros a recibir.

- Manejo de Errores:

  Implementa sistema de manejos de errores para la carga de imágenes y sonido dentro del juego.

- Separación de responsabilidades:

  Debido a que el código está compuesto por clases cada clase tiene su rol bien definido, lo que sigue el principio de separación de preocupaciones, aunque el código aún puede presentar mejoras en este aspecto.

- Uso de librerías externas:

  Presenta un buen manejo de librerías externas como pygame al manipular imágenes y sonido, además de implementar un interfaz gráfica para el juego.

- Adaptable:

Al definir y separar correctamente los parámetros del juego, es posible cambiar los valores dentro de este para ampliar la experiencia ya sea aumentando el tamaño o agregando una mayor cantidad de bombas.

- -Revisión de códigos en pares:

  El código fue revisado en conjunto con los integrantes del grupo donde se realizaron comentarios constructivos y un aprendizaje mutuo, además se llegó a una conformidad grupal de este.