Proyecto: Buscaminas

INFO299 – Arquitectura de software

Integrantes:
Luis Berrocal
Jaime González
Martín Suárez
Luciano Pavel

Equipo: 01
Fecha: 18 de diciembre de 2023

INTRODUCCIÓN

En el marco de este proyecto, se ha abordado el diseño e implementación de una versión del clásico juego “Buscaminas”, siguiendo una metodología de diseño sólida y aplicando buenas prácticas para garantizar la calidad del software resultante. El proceso del diseño ha sido guiado por la metodología 4+1, permitiendo una visión más completa de la arquitectura del software a través de diversas perspectivas. La implementación se llevará a cabo en el lenguaje de programación Python. Optamos por el uso de la librería Pygame para nuestro proyecto debido a su potencia en el desarrollo de juegos y aplicaciones multimedia, así como a su interfaz gráfica amigable.

Las reglas del juego a implementar son las mismas de siempre, el juego a implementar contará con un tablero rectangular, un cronómetro y un contador de minas. El objetivo será descubrir las casillas del tablero evitando presionar en las minas, utilizando la información por las casillas adyacentes con números.

A lo largo de este informe se detallarán los aspectos clave del diseño, incluyendo diagramas representativos de la arquitectura, patrones de diseño a seguir, etc. El uso de un repositorio github para facilitar la colaboración y versionado el código, proporcionando una mejor infraestructura para el desarrollo.

METODOLOGÍA

Metodología 4+1: En el proceso de desarrollo de este proyecto adoptamos la metodología 4+1 para garantizar una planificación estructurada y una composición integral del sistema. A continuación se explica cómo se aplicaron las distintas vistas arquitectónicas y los escenarios en nuestro proyecto:

Vista lógica: utilizamos el diagrama de clases para representar la estructura lógica del juego. Identificamos las clases principales, board, game_handler, display, entre otras, así como sus relaciones y métodos asociados. Esta vista nos permitió visualizar la organización de los componentes y sus interacciones desde una perspectiva funcional.

Vista de procesos: Empleamos el diagrama de secuencia para modelar la interacción dinámica entre las clases durante el desarrollo del juego. Estos diagramas capturaron el flujo de mensajes y la colaboración entre clases, proporcionando una visión detallada de cómo se ejecutan las acciones en respuesta a las interacciones del usuario.

Vista de desarrollo: Empleamos el diagrama de componentes para organizar y representar la estructura interna del código fuente. Este detalló la relación entre los distintos componentes del sistema, destacando cómo se agrupan en módulos y cómo interactúan entre sí durante la ejecución del juego.

Vista de física: el diagrama de despliegue se implementó para mostrar la distribución física del juego, identificando la relación entre los componentes y su ejecución en el entorno de ejecución, ya sea localmente o en un servidor.

Vista de casos de uso: Desarrollamos varios escenarios específicos que ilustraban el comportamiento del juego en situaciones clave, como la apertura de una celda minada o la victoria del jugador. Estos escenarios proporcionan casos de uso concretos que ayudaron a validar la arquitectura y a asegurar que el sistema cumplía con los requisitos funcionales y no funcionales.

Patrón de diseño "Game loop": La implementación del juego Buscaminas se fundamentó en la aplicación del patrón de diseño "Game loop", según lo detallado por Hans Schaa, Felipe Besoaín y Nicolas A. Barriga en su artículo "Introducción a patrones de diseño de software para el desarrollo de videojuegos" (2020), publicado en la revista Ludology [1]. Este enfoque nos permitió organizar de manera eficiente el ciclo de juego, garantizando una ejecución coherente de las acciones del jugador y proporcionando una experiencia de juego fluida.

Uso de buenas Prácticas:

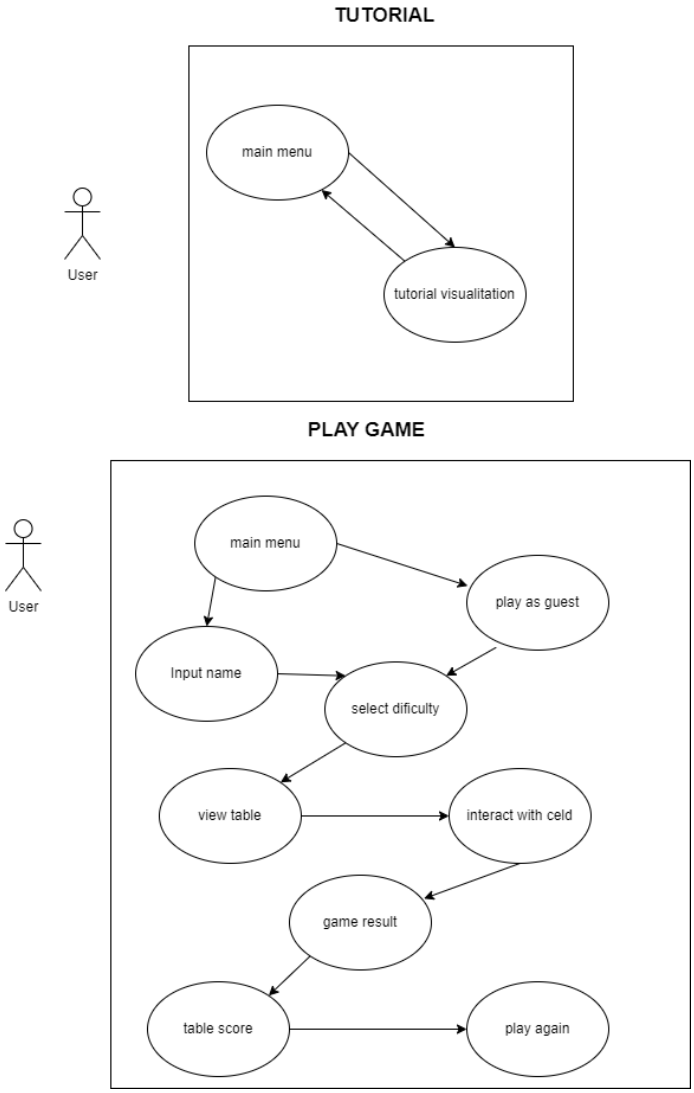
Modularización: Estructuramos el código en clases modulares, donde cada clase representa una entidad o funcionalidad específica del juego, como el tablero, celdas y la lógica del juego. Esta modularización favorece la legibilidad, facilita el mantenimiento y permite la reutilización de componentes.

Uso de diagramas: Integrando los diagramas mencionados anteriormente, visualizamos la estructura y la interacción entre las clases. Estos diagramas fueron cruciales para comprender la arquitectura del juego y coordinar eficientemente la implementación entre los miembros del equipo

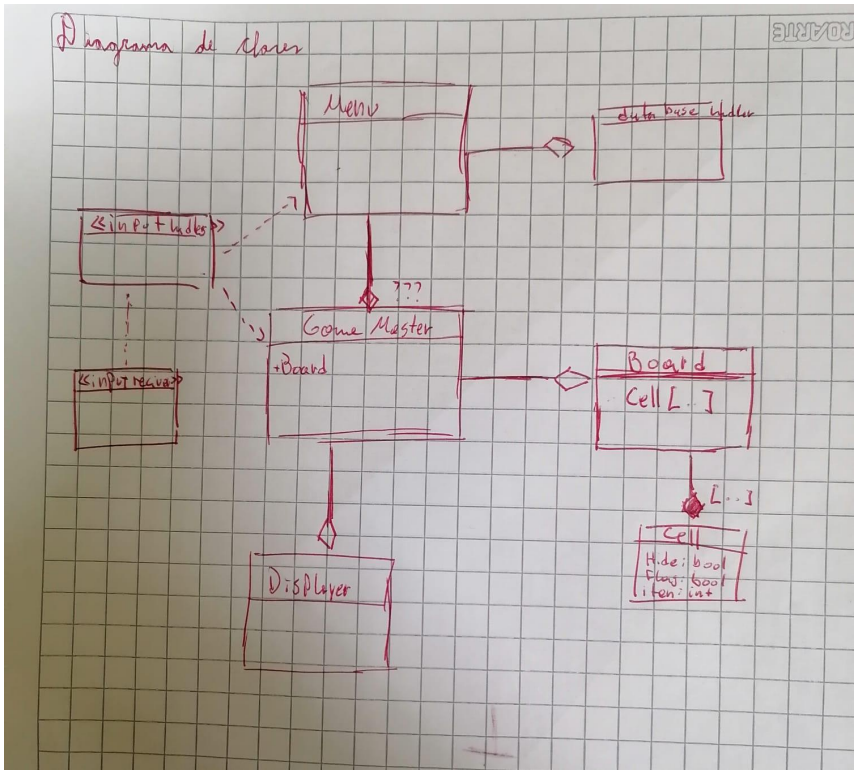
Uso de GitHub: Adoptamos GitHub como plataforma central para el desarrollo colaborativo del código fuente. La creación del repositorio, la gestión de ramas y las solicitudes de extracción facilitaron la colaboración entre los miembros del equipo. Además, el control de versiones permitió realizar un seguimiento preciso de los cambios y revertir modificaciones si fuera necesario.

RESULTADOS

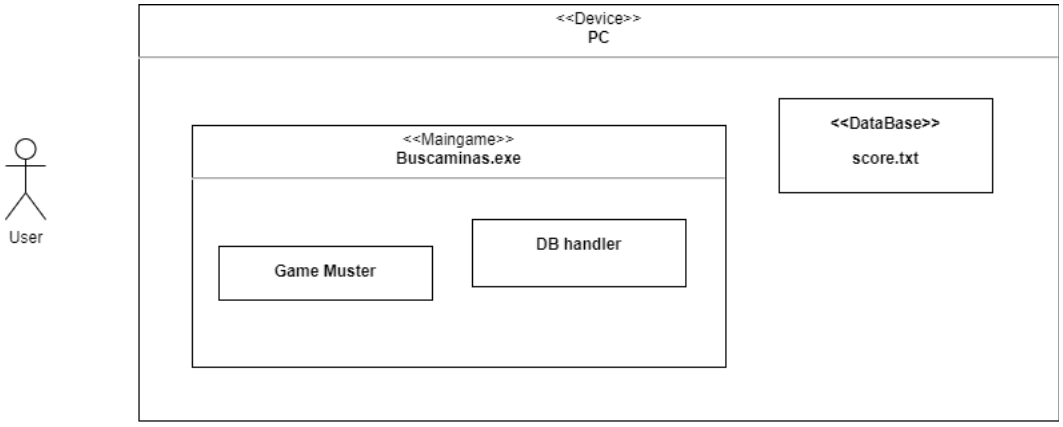
- **Diagrama de casos de uso:**



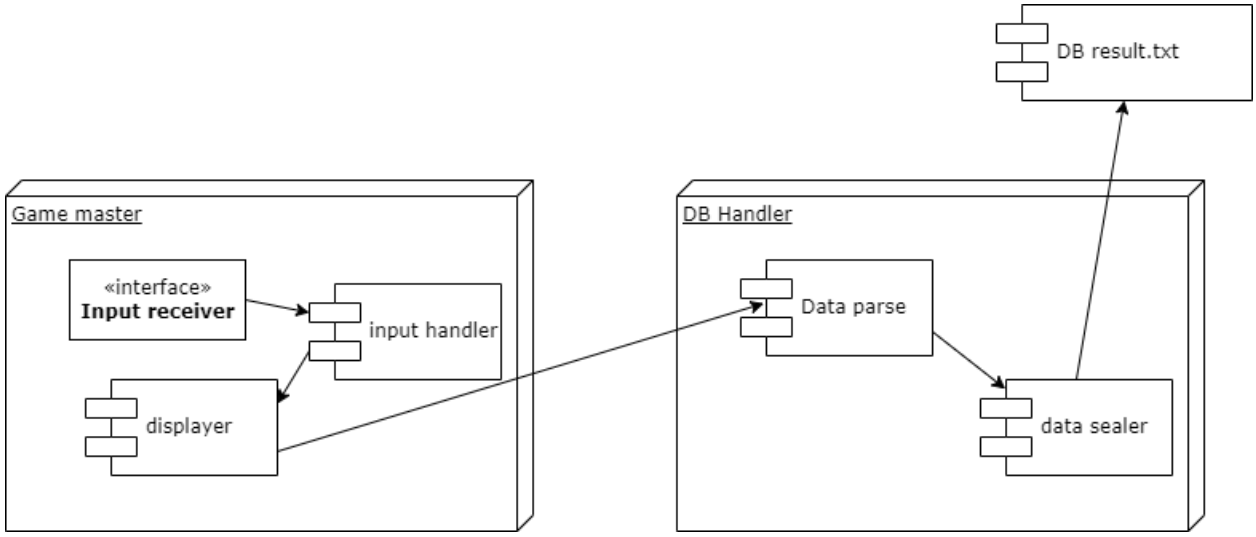
- **Diagrama de clases:**



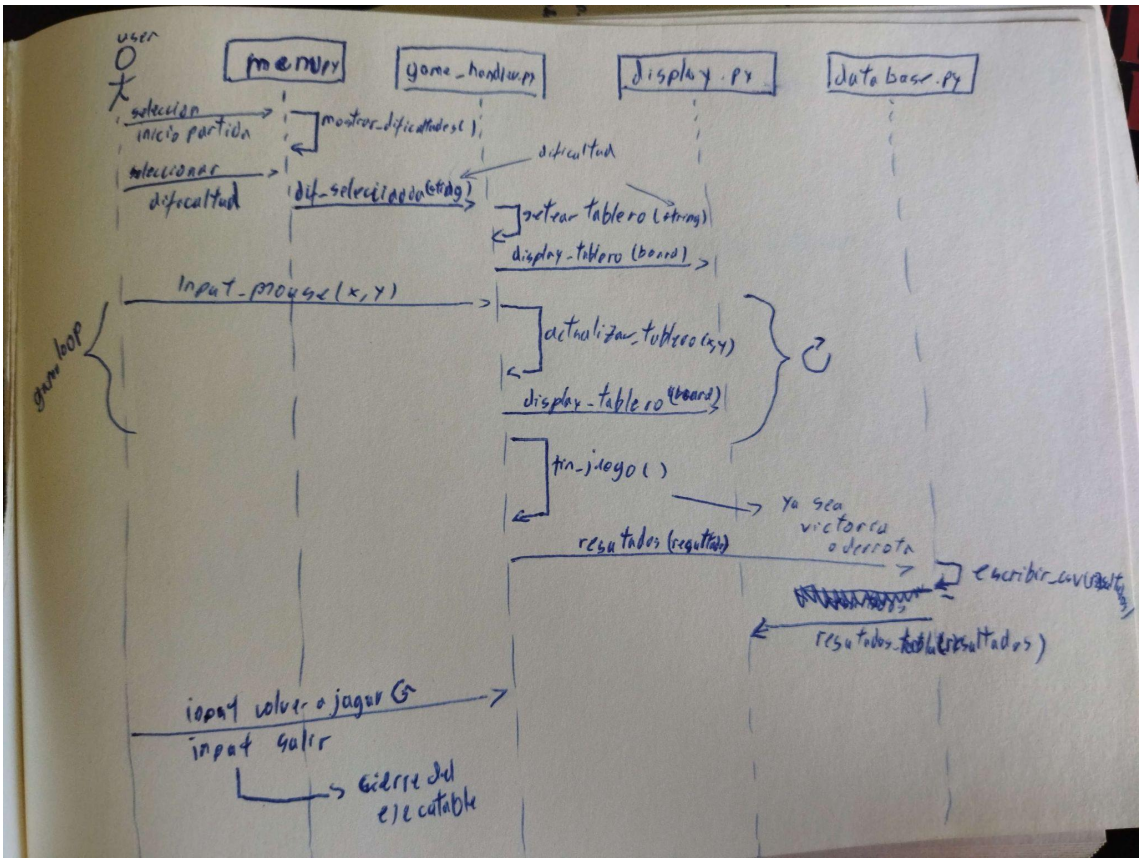
- **Diagrama de despliegue:**

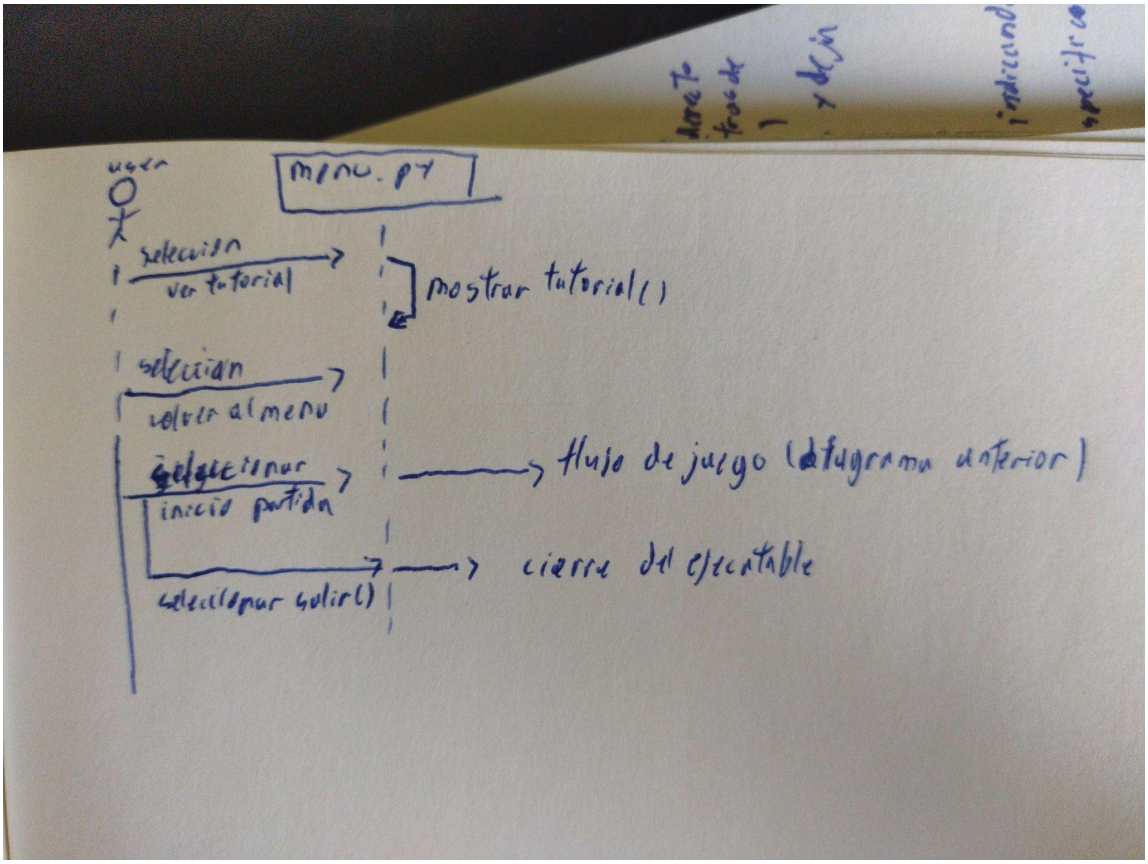


- **Diagrama de componentes:**



- **Diagramas de flujo:**





Aunque los diagramas proporcionaron una útil orientación para iniciar la redacción del código de manera modular y estructurada, en la práctica, al momento de redactar el código, nos permitimos agregar un par de clases adicionales (como board, input_handler, input_reciever, por ejemplo) y modificar los nombres de algunos métodos. Este enfoque dinámico surgió como una respuesta a las demandas específicas del desarrollo y las necesidades emergentes del proyecto.

Es importante destacar que no queremos decir con esto que los diagramas iniciales deberían ser ignorados. Más bien, los consideramos como un punto de partida sólido que proporciona una visión “general” del “arquetipo” de software que estamos implementando.

CONCLUSIONES

En el desarrollo del buscaminas que se nos ha planteado, hemos aplicado la metodología 4+1, un enfoque integral que combina las ventajas de diversas perspectivas para lograr un proceso de ingeniería de software robusto y eficiente.

Las principales ventajas que trajo este enfoque de trabajo fueron las siguientes:

Claridad en la Comunicación: La metodología 4+1 proporcionó un marco claro y estructurado de trabajo entre los miembros del equipo, bien sea encargados del diseño de la arquitectura de software o bien de la codificación esto nos permite tener una mayor cohesión como equipo y finalmente una entrega más óptima respecto a otras implementaciones.

Gestión Efectiva de Requisitos: La perspectiva de casos de uso (escenarios) dentro de la metodología 4+1 nos permitió centrarnos en las necesidades del jugador. Esto garantizó que los requisitos del juego fueran claramente definidos y comprendidos.

Diseño Modular y Reutilización de Componentes: La vista de diseño permitió una división clara del sistema en módulos independientes, fomentando la reutilización de código y facilitando la mantenibilidad. Esta modularidad se tradujo en un desarrollo más eficiente, ya que cada equipo pudo trabajar en paralelo en diferentes partes del juego.

Alineación con Prácticas Ágiles: si bien no fue ocupada en este caso, podemos prever cómo fácilmente esta metodología 4+1 se podría integrar eficazmente con prácticas ágiles, permitiendo iteraciones rápidas y flexibilidad en la adaptación a cambios en los requisitos.

ANEXOS:

[1] Schaa, H., Besoaín, F., & Barriga, N. A. (2020). Introducción a patrones de diseño de software para el desarrollo de videojuegos. Ludology, 2, 30-38. Recuperado de INTRODUCCIÓN A PATRONES DE DISEÑO DE SOFTWARE PARA EL DESARROLLO DE VIDEOJUEGOS

[2] Pygame documentation: Pygame Front Page — pygame v2.6.0 documentation

[3] Minesweeper Rules: How To Play Minesweeper