Instituto de Informática
Facultad de Ciencias de la Ingeniería
Universidad Austral de Chile
INFO229 Arquitectura de Software

Proyecto de arquitectura de software: Buscaminas

Integrantes: Cristóbal Rebolledo O.
Carolina Obreque H.
Benjamín Parra B.
Profesor: Matthieu Vernier
Fecha de entrega: Lunes 18 de diciembre de 2023
Valdivia, Chile

Resumen

Este informe presenta la implementación de un juego de Buscaminas utilizando Python y la interfaz gráfica de Tkinter. La arquitectura del software se diseñó siguiendo la metodología 4+1, aplicando buenas prácticas y patrones de diseño para mejorar la calidad del software.

Presentamos una estrategia peculiar para imaginar el funcionamiento del buscaminas, basándose en nodos, de manera de reducir el acoplamiento al mínimo con el mapa para delegar a las celdas la responsabilidad de descubrir a sus hermanas si así se indica. Esta radica en un diseño más simple pero complejiza la parte de llevar a cabo el seguimiento de los estados del juego, el cual debe poderse ganar o perder.

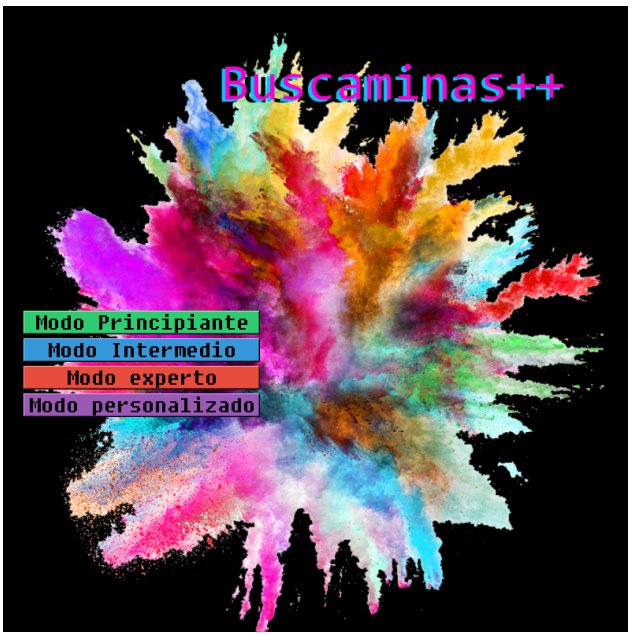

Figura 1: Menú de inicio del juego

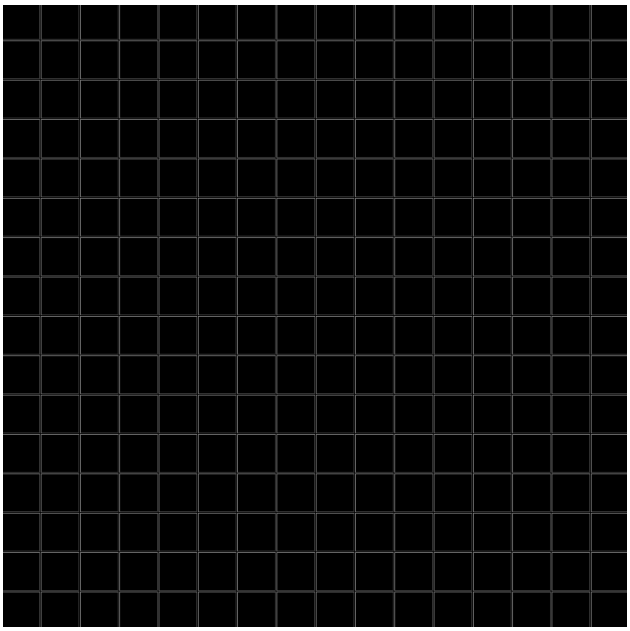

Figura 2: Mapa del juego sin descubrir

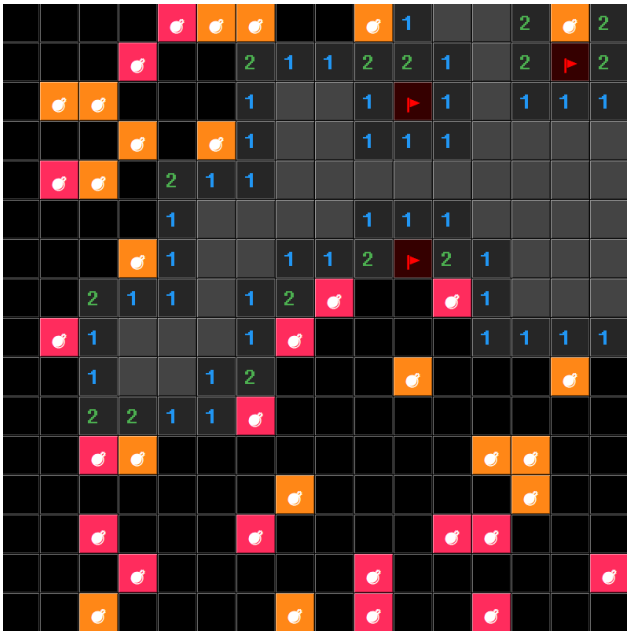

Figura 3: Ejemplo de perder partida

Índice de Contenidos

1. Introducción **1**
1.1. Decisiones de plataforma 1
1.2. Reactividad de los componentes 1
1.3. Metodologia 4+1 1

2. Metodología 4+1 **2**
2.1. Vista de casos de uso 2
2.2. Vista Lógica 3
2.2.1. Celda 3
2.3. Vista de Desarrollo 4
2.4. Vista de Procesos 4
2.5. Vista Física 7

3. Desarrollo **8**
3.1. Implementación en python 8
3.2. Uso de patrones de diseño 8
3.3. Prácticas utilizadas 8
3.4. Lo que no se implementó 9
3.5. Despliegue de software 9

4. Conclusión **10**

5. Instrucciones para el usuario **10**

Lista de Figuras

1. Menú de inicio del juego i
2. Mapa del juego sin descubrir ii
3. Ejemplo de perder partida ii
4. Diagrama de casos de uso 2
5. Diagrama de clases 3
6. Diagrama de componentes 4
7. Diagrama de secuencia caso de uso: configurar partida 5
8. Diagrama de secuencia caso de uso: descubrir celda 5
9. Diagrama de secuencia caso de uso: colocar bandera 6
10. Diagrama de secuencia caso de uso: quitar bandera 6
11. Diagrama de despliegue 7

1. Introducción

El objetivo del proyecto fue implementar una arquitectura de software utilizando la metodología 4+1. Se aplicó la metodología para diseñar vistas lógicas, de procesos, de desarrollo y físicas, junto con un conjunto de casos de uso para comprender y diseñar el sistema de manera integral. En el presente informe se mostrarán las técnicas utilizadas para el desarrollo del conocido juego "Buscaminas". Se intentó, además, llevar a cabo un enfoque diferente en el diseño del juego, para así poder abrir más el debate acerca de las decisiones de diseño.

1.1. Decisiones de plataforma

Se eligió el módulo Tkinter de Python para realizar la construcción del programa, ya que permite representar los objetos como si fueran elementos visuales fácilmente. Esto implica que es amigable con el diseño lógico de la aplicación, ya que permite una conversión directa de objeto a su representación gráfica interactiva.

1.2. Reactividad de los componentes

El diseño inicial toma en consideración tres componentes principales, los cuales son las celdas, el mapa y el estado de juego, con estas tres cosas y un frame se puede representar bien lo que se quiere mostrar. El juego debe manejar cambios en el estado, lanzados por presionar las celdas. Se optó por colocar las celdas como una red, de esta manera no se necesita una matriz para almacenar los botones Celda, además de permitir un diseño único y reactivo. Para la interacción entonces es necesario que cada celda conozca sus hermanos para construir la red. El mapa se puede encargar de construir la red de nodos y juntarlos en un frame. Gracias a la independencia de los botones, no se necesita un objeto intermediario, por lo que puede traer beneficios para la simplicidad del diseño, a costa de añadir complejidad ya que implica construir la red en lugar de usar un simple arreglo plano o una matriz. Para otros detalles de reactividad, para mantener el paralelismo de los componentes se utilizarán threads. Algunos ejemplos son que cambien de color las bombas al explotar, o mantener el timer (descartado del codigo).

1.3. Metodologia 4+1

Se adoptó la metodología 4+1 para obtener una comprensión completa y sistemática del sistema. Las diferentes vistas, como casos de uso, lógica, desarrollo, procesos y física, se emplearon para describir los distintos aspectos del software. La vista de casos de uso detalla las interacciones clave entre el jugador y el sistema, mientras que la vista lógica representa la jerarquía de los objetos y sus relaciones. La vista de desarrollo ofrece una visión estructural de la implementación, la vista de procesos comprende el flujo de ejecución, y la vista física ilustra la distribución espacial de los componentes.

2. Metodología 4+1

2.1. Vista de casos de uso

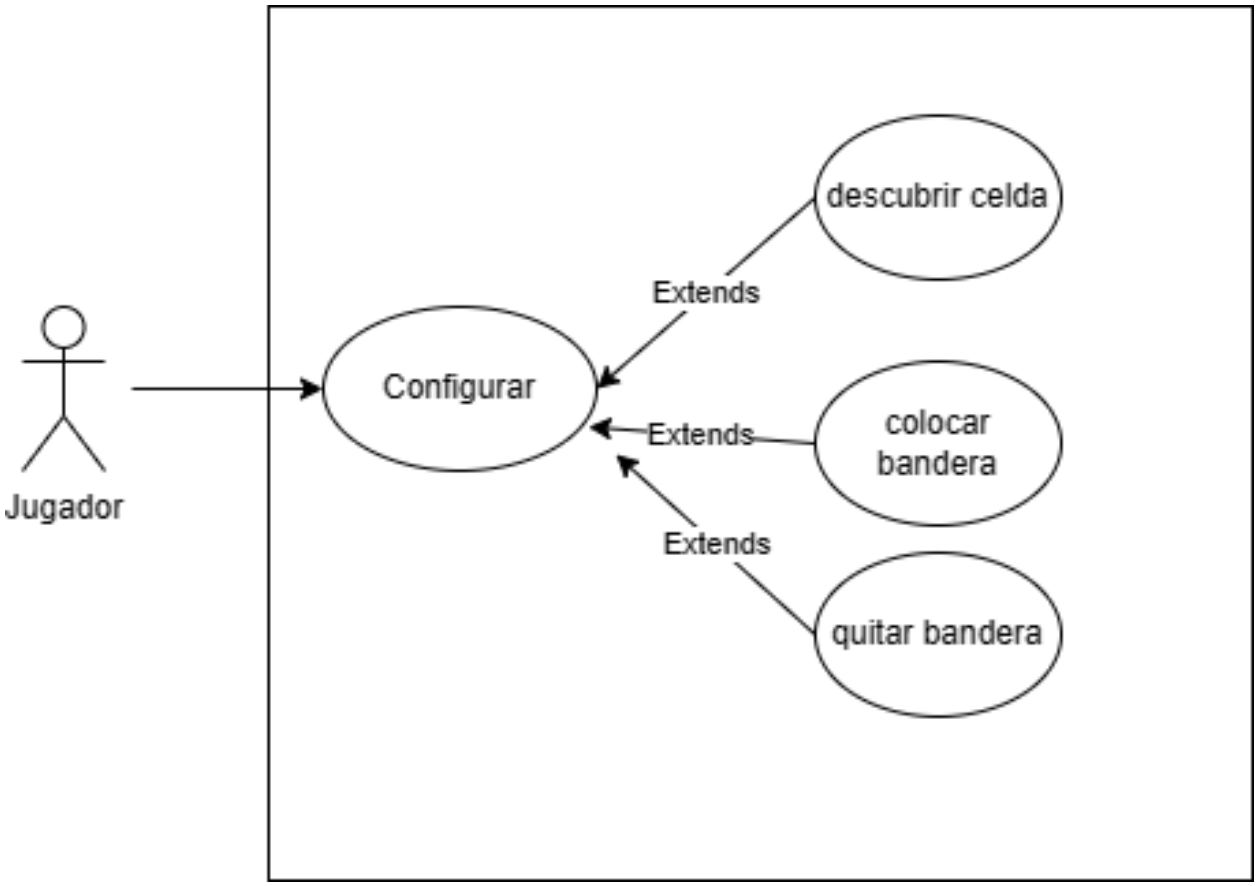

Figura 4: Diagrama de casos de uso

Se presenta un diagrama de casos de uso con un único actor, el Jugador. Este puede configurar la partida y realizar clicks derecho o izquierdo para interactuar con el juego, estas interraciónes lo llevaran a ganar o perder la partida.

2.2. Vista Lógica

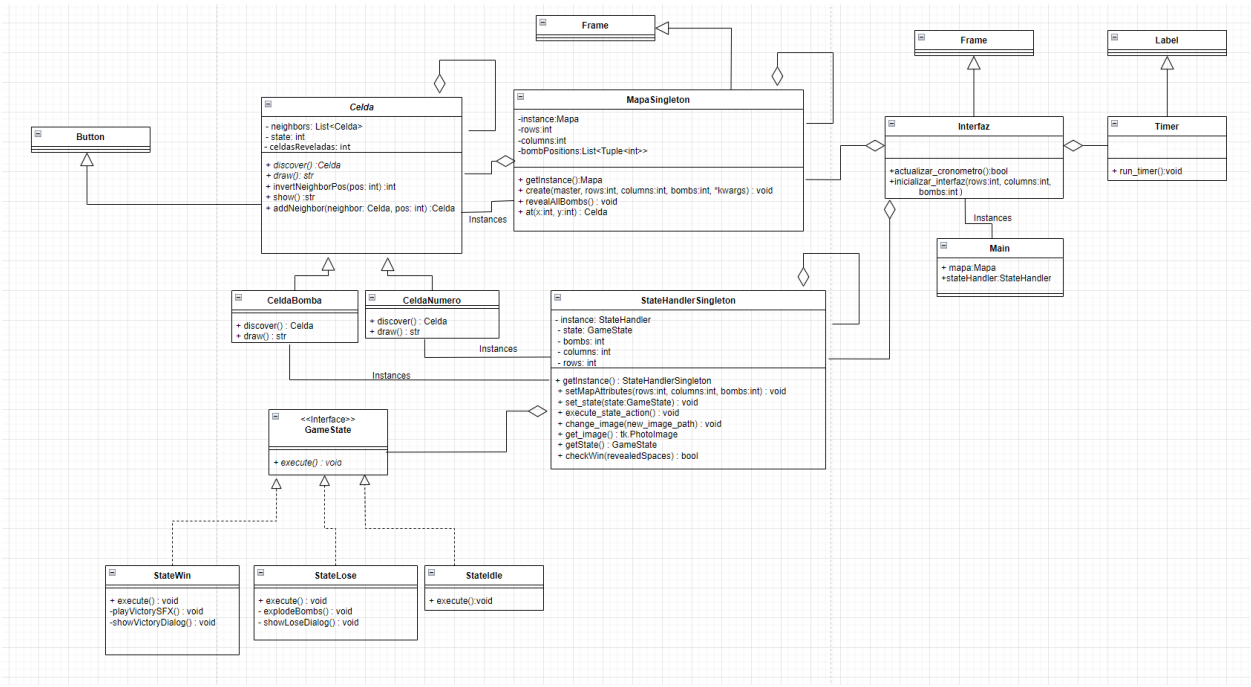

Figura 5: Diagrama de clases

Se diseñó un diagrama de clases que representa la jerarquía de las celdas del juego, incluyendo celdas normales, con bomba y numeradas. Se aplicaron conceptos de herencia y abstracción para modelar la relación entre estas clases. También se utiliza el patrón de diseño State para manejar el estado del juego.

2.2.1. Celda

Esta es una clase abstracta que define a una Celda. Conoce a todos sus vecinos y los tiene en un arreglo. Extiende a tkinter.Button Puede ser CeldaNumero o CeldaBomba, y cada una conoce el comportamiento que debe llevar al ser descubierta, sin embargo, necesita de metodos para tratar los estados sobre los que se encuentra, los cuales pueden ser MARCADA, NO-MARCADA y REVELADA. De esta manera, la clase Celda contiene las funciones necesarias para evitar que cada clase heredada tenga que implementar las soluciones por su cuenta. Algunas funciones interesantes cuyas diferencias son importantes a notar son:

- discover: Funcion abstracta que describe el comportamiento de descubrir la celda. Se retorna la misma celda
- draw: Funcion abstracta, debe devolver la conversion a string de la celda independientemente de su estado.
- invertNeighborPos: Calcula la posicion opuesta en el arreglo de vecinos

- show: Devuelve el str que corresponde al valor de la celda considerando el estado en el que está
- addNeighbor: Agrega un vecino a la celda
- flag: Se marca o desmarca la celda

2.3. Vista de Desarrollo

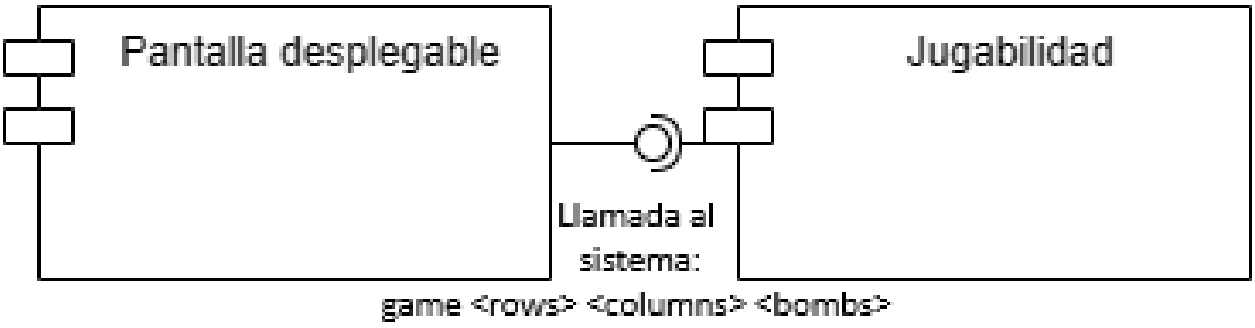

Figura 6: Diagrama de componentes

Los principales componentes usados son la pantalla desplegable, que incluye el menú y la interfaz inicial, y la Jugabilidad que es el funcionamiento principal del juego. Dentro de este ultimo se encuentra el diagrama de clases mostrado. Hay que recordar que la interfaz del menu, es aparte de la interfaz de la pantalla de juego.

2.4. Vista de Procesos

A continuación se presentan los diagramas de secuencia considerando los casos de uso dados anteriormente:

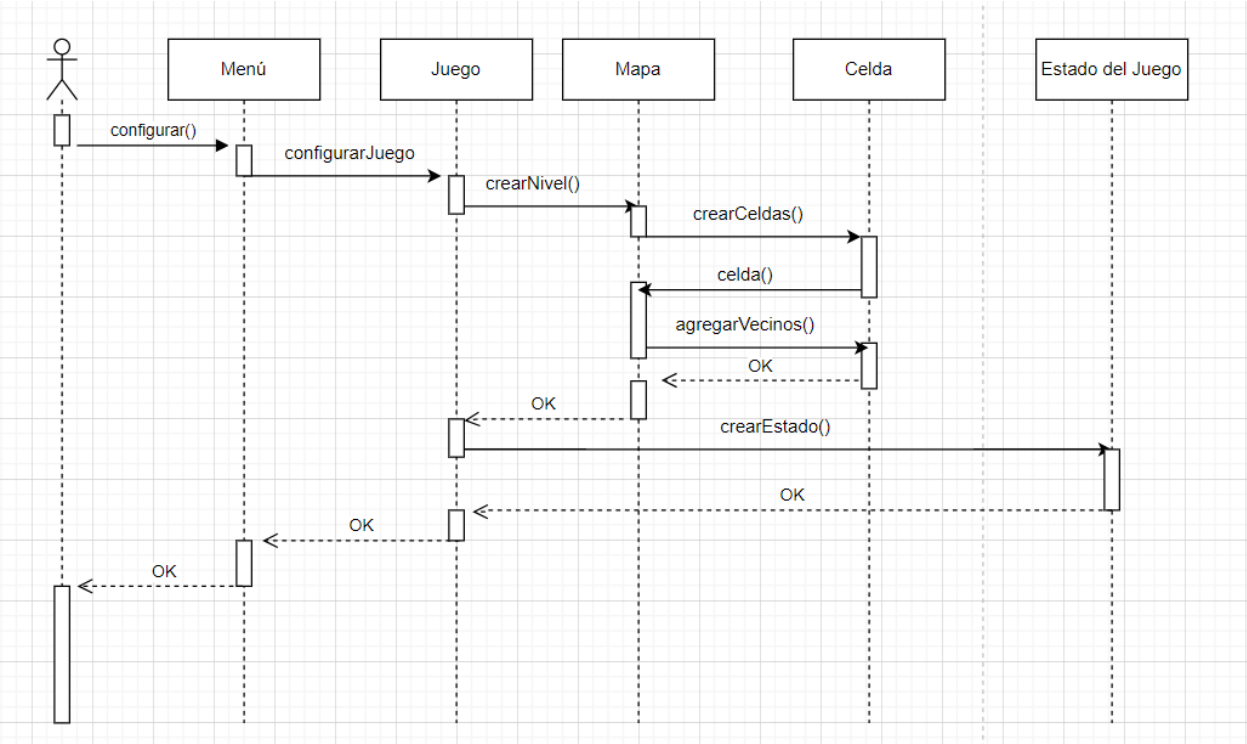

Figura 7: Diagrama de secuencia caso de uso: configurar partida

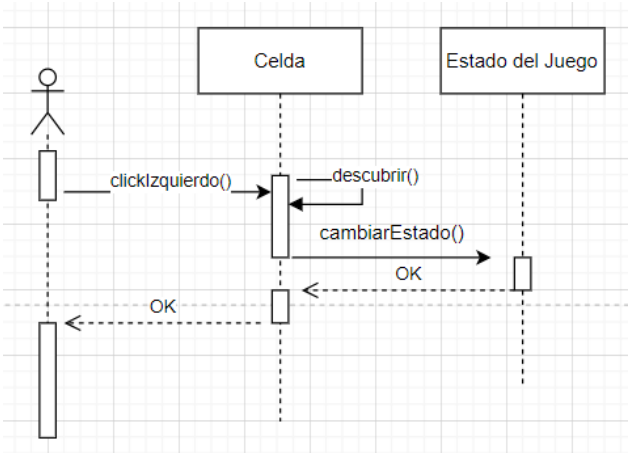

Figura 8: Diagrama de secuencia caso de uso: descubrir celda

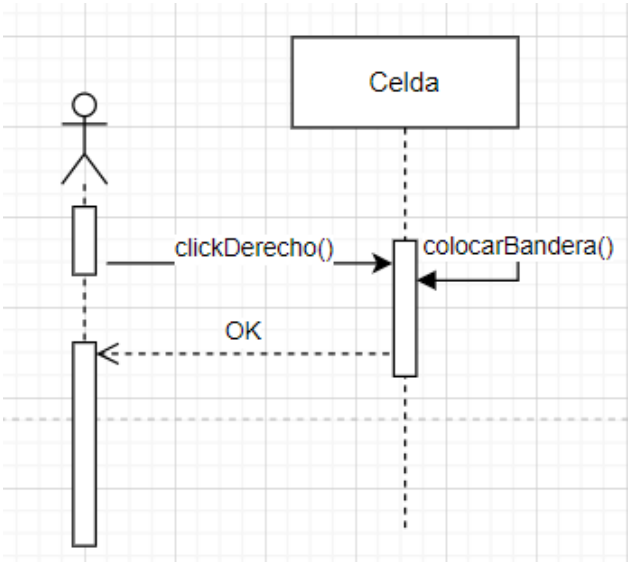

Figura 9: Diagrama de secuencia caso de uso: colocar bandera

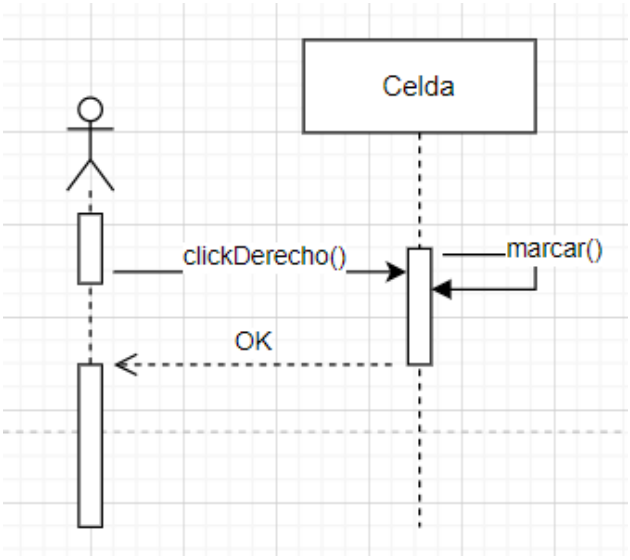

Figura 10: Diagrama de secuencia caso de uso: quitar bandera

2.5. Vista Física

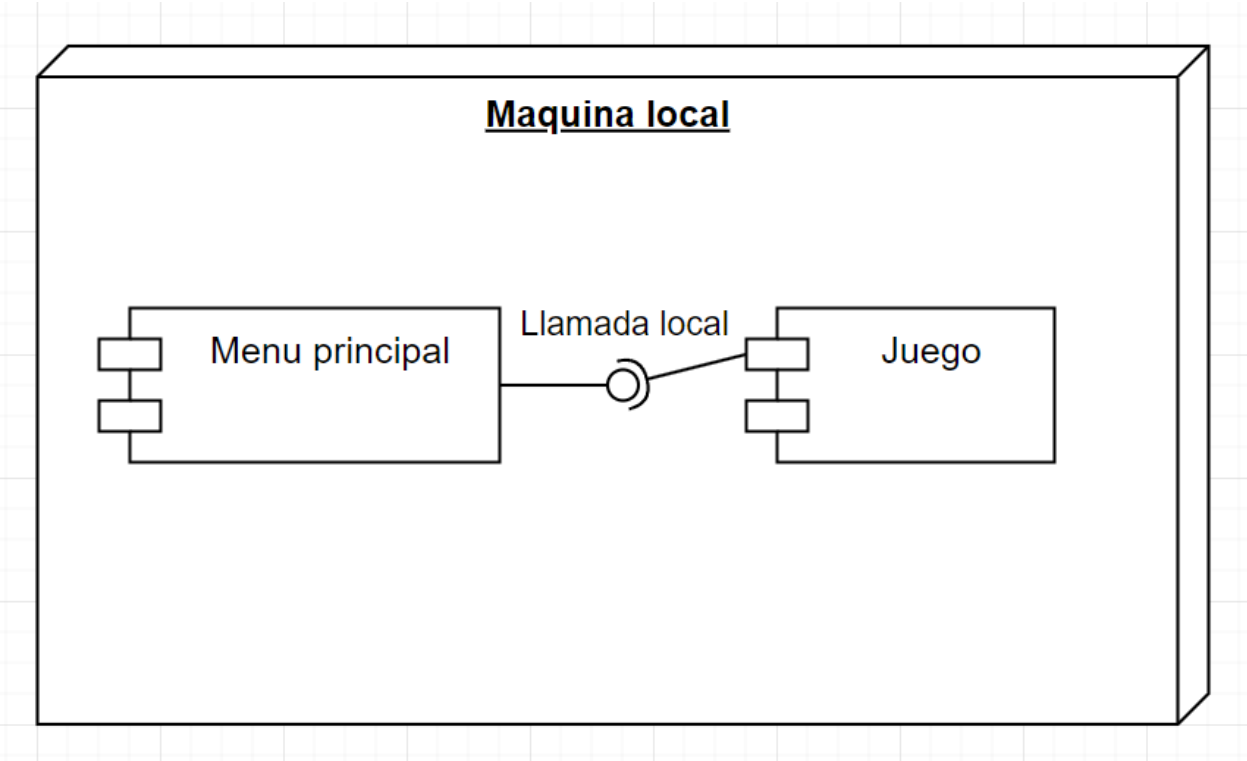

Figura 11: Diagrama de despliegue

Se creó una interfaz gráfica utilizando Tkinter para representar el mapa del juego. El juego inicia con el menú para luego crearse el mapa que esta compuesto de celdas. Las celdas se dispusieron en una cuadrícula, y se configuraron eventos de click para interactuar con las celdas. Si bien esta aplicacion esta pensada para montar en el mismo equipo, si la llamada por consola puede ser realizada a distancia nada evita que el

3. Desarrollo

3.1. Implementación en python

Se implementó el juego de Buscaminas utilizando el lenguaje de programación Python y la biblioteca gráfica Tkinter para la interfaz de usuario. La ejecución y despliegue del software involucran los siguientes aspectos:

Python se eligió como el lenguaje principal para la implementación debido a su simplicidad, legibilidad y versatilidad. La programación orientada a objetos se utilizó extensivamente para modelar las celdas y la lógica del juego.

3.2. Uso de patrones de diseño

- Se implementó el patrón de diseño Singleton para garantizar una única instancia del mapa del juego.
- Se implementaron los patrones de diseño Singleton y State en StateHandler para controlar los estados del juego y que estos estén en una sola instancia.

3.3. Prácticas utilizadas

La biblioteca Tkinter proporcionó la capacidad de manejar eventos de interfaz de usuario, como clics izquierdos y derechos en las celdas. Estos eventos desencadenaron las acciones correspondientes en el juego, como descubrir una celda o marcarla con una bandera.

La jerarquía de clases se diseñó cuidadosamente para aprovechar la herencia y la abstracción. Las clases Celda, CeldaBomba y CeldaNumero formaron la base del juego, permitiendo una estructura flexible y reutilizable.

La modularidad y la claridad en la estructura del código facilitaron la reutilización de componentes en el desarrollo. Las celdas y sus comportamientos se implementaron de manera que pudieran adaptarse fácilmente a cambios futuros o extensiones.

El diseño de red de las celdas tuvo muchos beneficios y desventajas; por ejemplo, si bien redujeron las responsabilidades del mapa al mínimo, tuvieron que ser ellas las que manejaron el estado de juego, por lo que esa parte puede ser mejorable. Quedará a la tarea del lector ver si encuentra una manera de mejorar este apartado, viendo el enfoque lógico.

El enfoque en la documentación clara es esencial y, al continuar proporcionando comentarios detallados, se facilitará la comprensión y el mantenimiento del código en el futuro. La flexibilidad y reutilización de componentes son evidentes en el diseño, lo cual es fundamental para adaptarse a posibles cambios y extensiones. Lo mas importante de la documentacion, sin lugar a duda, es la claridad de explicación que otorga, permitiendo a los demas desarrolladores entender rapidamente el codigo, lo que mejoró la comunicación en el equipo.

3.4. Lo que no se implementó

Aunque el juego está funcional, no logramos implementar completamente la interfaz que incluye el estado del juego, el mapa y el temporizador. Esto se debió a numerosas dificultades para integrar estas clases de manera efectiva y garantizar el correcto funcionamiento del juego. Como resultado, se tomó la decisión de dejar la interfaz únicamente en los diagramas, ya que la implementación práctica presentaba desafíos significativos. Pero para lograr la interfaz simplificada, su usó el main para instanciar las clases Mapa y StatleHandler. Para mejorar el diseño, se podría contemplar la posibilidad de almacenar las celdas reveladas en una ubicación diferente a la clase Çelda", quizás en un objeto separado como el ya conocido "Mapa"

3.5. Despliegue de software

- La implementación del juego de Buscaminas se despliega en un entorno Python utilizando la interfaz gráfica de Tkinter.
- El proceso de despliegue implica la ejecución del programa, donde la interfaz gráfica se genera mediante Tkinter, permitiendo al usuario interactuar con el juego.
- La elección de Tkinter como plataforma de desarrollo garantiza una representación gráfica intuitiva y amigable, facilitando la experiencia del usuario. Se recomienda seguir las instrucciones proporcionadas para una experiencia óptima al jugar al Buscaminas.

4. Conclusión

La implementación del juego Buscaminas siguiendo la metodología 4+1 y aplicando patrones de diseño y permitió desarrollar un software estructurado y eficiente. A continuación, se destacan algunas conclusiones clave:

- Metodología 4+1: La aplicación de esta metodología proporcionó una comprensión completa y detallada de la arquitectura del software. Las distintas vistas facilitaron el diseño integral del sistema y la identificación de los aspectos clave de la implementación.
- Vistas Específicas: Cada vista proporcionó información valiosa. La vista de casos de uso ayudó a definir las interacciones del jugador, mientras que la vista lógica guió la implementación de las clases y la relación entre ellas. Las vistas de procesos y desarrollo fueron esenciales para comprender la ejecución del software y su estructura interna.
- Uso de Patrones de Diseño: La aplicación de patrones de diseño, como Singleton y State, mejoró la eficiencia y la claridad del código. Singleton garantizó la existencia de una única instancia del mapa del juego, y State permitió gestionar los diferentes estados de las celdas.
- Interfaz Gráfica con Tkinter: La implementación de la interfaz gráfica con Tkinter facilitó la representación visual del juego. La disposición en cuadrícula y la gestión de eventos proporcionaron una interacción fluida y amigable con el usuario.
- Diseño único: Permitió expresar un pensamiento diferente para otorgar nuevas perspectivas sobre el desarrollo del juego.

5. Instrucciones para el usuario

Es necesario tener Python (versión recomendada: 1.12.1) y pygame instalados. Es recomendado tener un charset con emojis, y una pantalla de resolución al menos hd. Luego debes iniciar el main.py de la carpeta base del proyecto. El readme tiene informacion mas detallada.
Para jugar Buscaminas, sigue estos pasos:

- Selecciona el modo de juego (Principiante, Intermedio, Experto o Personalizado) desde el menú principal.
- En el modo personalizado, configura el número de filas, columnas y minas.
- Haz click izquierdo en una celda para descubrirla si no estaba descubierta.
- Haz click derecho en una celda para marcarla o desmarcarla con una bandera, asi evitarás descubrirla por error.
- El juego termina cuando no quedan espacios vacios sin marcar o se descubre una mina.

¡Disfruta jugando Buscaminas!