

**Universidad Francisco José de Caldas**

**Manual técnico**

**Juego Buscaminas**

Desarrollado en java



David Santiago Buitrago C. 20221020085

Henry Ricaurte 20221020084

Julian Andres Vargas M. 20221020069

Bogotá 9 de septiembre 2023

Tabla de contenido

**Escribir el título del capítulo (nivel 1)1**

Escribir el título del capítulo (nivel 2)2

Escribir el título del capítulo (nivel 3)3

**Escribir el título del capítulo (nivel 1)4**

Escribir el título del capítulo (nivel 2)5

Escribir el título del capítulo (nivel 3)6

# Introducción

## Objetivo

El buscaminas es un juego de computadora clásico que tiene como objetivo principal proporcionar entretenimiento y desafío mental a los jugadores. Su función principal es crear un rompecabezas lógico en forma de un campo minado, donde los jugadores deben despejar el terreno y encontrar todas las casillas vacías sin detonar ninguna mina.

El propósito de la aplicación del buscaminas es brindar a los usuarios una experiencia divertida y estimulante que requiera el uso de la lógica y la deducción. Además de entretener, este juego también puede ayudar a mejorar la capacidad de resolución de problemas y la toma de decisiones, ya que los jugadores deben tomar decisiones estratégicas sobre qué casillas abrir y cuáles evitar, basándose en la información disponible.

## Audiencia

El manual técnico del juego Buscaminas está escrito pensando en un público técnico, centrándose en particular en los ingenieros de sistemas y los profesores. Está destinado a personas con conocimientos técnicos y conocimientos fundamentales de principios de informática y programación. El objetivo principal de esta guía es brindar a los jugadores instrucciones completas sobre cómo jugar Buscaminas, incluidas reglas del juego, consejos para ganar y técnicas efectivas para resolver acertijos.Si los profesores de ingeniería de sistemas quieren utilizar el juego como recurso didáctico en sus clases, puede haber información adicional en el manual que deben conocer. Esto podría incluir ideas para incorporar juegos en lecciones sobre razonamiento, toma de decisiones y resolución de problemas en el contexto de la informática.

# Marco teórico

# Descripción General de la Aplicación

## Funcionalidades clave.

* **Tablero de juego:** La aplicación debe proporcionar un tablero de juego que consta de un campo cuadriculado, donde se encuentran las casillas del juego.
* **Minas:** Se deben colocar minas de forma aleatoria en el tablero antes de que comience el juego. El número de minas y su ubicación también puede variar según la dificultad seleccionada.
* **Casillas:** Cada casilla en el tablero puede estar en uno de los siguientes estados:
  + Cubierta: El jugador aún no ha interactuado con ella.
  + Descubierto: El jugador ha hecho clic en la casilla y se revela su contenido.
  + Marcada: El jugador ha marcado una casilla sospechosa de contener una mina con una bandera o un símbolo equivalente.
* **Contenido de las casillas:** Las casillas pueden contener una mina (en cuyo caso el juego termina si se descubre), un número que indica cuántas minas están adyacentes a esa casilla o están vacías.
* **Interacción del jugador:** Los jugadores deben poder hacer clic en las casillas para descubrir su contenido o marcarlas como sospechosas de contener una mina. También deben poder deshacer acciones si cometen un error.
* **Reglas del juego:** La aplicación debe seguir las reglas estándar del Buscaminas, como el hecho de que si se descubre una casilla vacía, se revelen automáticamente todas las casillas adyacentes vacías hasta que se alcance una casilla con números.
* **Dificultad ajustable:** Los jugadores deben poder elegir entre diferentes niveles de dificultad que afecten al tamaño del tablero y la cantidad de minas presentes.
* **Personalización**: La aplicación puede permitir a los jugadores personalizar el aspecto del juego, como el diseño del tablero y los iconos utilizados para las minas y las casillas marcadas.

Estas son las funcionalidades clave que se esperan en una aplicación de Buscaminas típica, aunque algunas versiones pueden agregar características adicionales para aumentar la diversión y el desafío.

**3.2 Diagrama UML**

******

**3.3 Arquitectura**

La arquitectura empleada para desarrollar el juego de Buscaminas en Java sigue un enfoque orientado a objetos y está organizada en varias clases que se encargan de diferentes aspectos del juego.

La arquitectura del programa se podría dividir en 3 carpetas: launcher, lógica y presentación:

***Package“Launcher”***

* **Clase Launcher:**

1. **Responsabilidades:**
   1. Iniciar la aplicación y configurar su modelo.
2. **Atributos:**
   1. ***private Modelo m:*** Este atributo es una referencia a un objeto de la clase "Modelo". Es privado, lo que significa que sólo es accesible dentro de la clase "Launcher".
3. **Métodos:**
   1. ***public Launcher( ):*** Este es el constructor de la clase "Launcher". Se encarga de crear una instancia del modelo (Modelo m = new Modelo()) y luego llama al método iniciar() del objeto del modelo recién creado. Este constructor se utiliza para inicializar la aplicación.}}
   2. ***public static void main(String[ ] args):*** Este es el método principal de la clase "Launcher". Es el punto de entrada de la aplicación. Cuando se ejecuta la aplicación, se crea una instancia de "Launcher" en este método (con new Launcher()) y, como resultado, se inicia la aplicación llamando al constructor. Este método es estático, lo que significa que pertenece a la clase en lugar de a una instancia específica de la clase.

En resumen, la clase "Launcher" se utiliza para iniciar la aplicación creando una instancia de la clase "Modelo" y llamando a su método "iniciar()". El método main se encarga de crear una instancia de "Launcher" para comenzar la aplicación cuando se ejecuta.

***Package “Lógica”***

* **Clase LogicaMinas**

1. **Responsabilidades:**
   1. La clase "LogicaMinas" se encarga de gestionar la lógica relacionada con la explosión de las minas en el juego de Buscaminas.
   2. Tiene un constructor que recibe un objeto de la clase "Tablero" y llama al constructor de la clase base "LogicaTablero" con ese tablero.
   3. Proporciona un método "Explotar" que explota las minas en el tablero visualmente. Cambia el color de los paneles que representan las minas a un color específico y utiliza un hilo para realizar estas actualizaciones visuales con un cierto retraso.
2. **Atributos:**
   1. No tiene atributos propios, pero hereda los atributos de la clase base "LogicaTablero" a través de la llamada al constructor superior.
3. **Métodos:**
   1. ***public LogicaMinas(Tablero tablero):*** Este es el constructor de la clase "LogicaMinas". Recibe un objeto "Tablero" como parámetro y llama al constructor de la clase base "LogicaTablero" pasando ese tablero. Esto permite que la clase "LogicaMinas" trabaje con el mismo tablero que se utiliza en el juego.
   2. ***public void Explotar(JPanel panel, ArrayList<JPanel> pMinas):*** Este método se encarga de explotar las minas visualmente en el tablero. Recibe un panel y una lista de paneles que representan las minas. Utiliza un hilo para cambiar el color de los paneles de las minas a un color específico, creando una animación de explosión.
   3. ***private ArrayList<Integer> listaMinas():*** Este método privado se utiliza para crear una lista de enteros que representan los paneles que contienen minas en el tablero. Itera a través de las celdas del tablero y agrega las coordenadas de los paneles que contienen minas a esta lista.

En resumen, la clase "LogicaMinas" se encarga de gestionar la lógica relacionada con la explosión de las minas en el juego de Buscaminas, incluyendo la representación visual de la explosión. Utiliza un tablero proporcionado por su constructor y contiene métodos para identificar las posiciones de las minas en el tablero y para realizar la explosión visualmente.

* **Clase LogicaTableros**

1. **Responsabilidades**
   1. La clase "LogicaTablero" se encarga de gestionar la lógica relacionada con la manipulación del tablero del juego.
   2. Proporciona métodos para verificar si una celda contiene una mina, para colocar una banderita en una celda, para contar las minas adyacentes a una celda y para plantar y expandir áreas del tablero cuando se hace clic en una celda vacía.
   3. También proporciona métodos auxiliares para convertir coordenadas de matriz en un número de panel y para colorear celdas del tablero.
2. **Atributos:**
   1. ***private Tablero tablero:*** Un atributo que almacena una referencia al objeto de la clase "Tablero", que representa la estructura del tablero de juego.
3. **Métodos:**
   1. ***public LogicaTablero(Tablero tablero):*** Constructor que recibe un objeto "Tablero" como parámetro y establece la referencia al tablero. Se utiliza para inicializar la lógica del tablero con un tablero específico.
   2. ***public Tablero getTablero():*** Método que devuelve el objeto "Tablero" asociado a esta lógica.
   3. ***public void setTablero(Tablero tablero):***Método que permite cambiar el objeto "Tablero" asociado a esta lógica.
   4. ***public boolean verMina(int fila, int columna):*** *Método que verifica si una celda en la fila y columna especificadas contiene una mina en función del estado de esa celda en el tablero.*
   5. ***public void Banderita(JPanel panel):*** Método que coloca una banderita en un panel específico si el panel no tiene un color de fondo correspondiente a una mina o a una celda abierta.
   6. ***public int mirarAlrededor(int fila, int columna):*** Método que cuenta las minas adyacentes a una celda en las coordenadas especificadas. Utiliza un arreglo de desplazamiento para verificar las celdas adyacentes.
   7. ***private boolean esValida(int fila, int columna):*** Método privado que verifica si una celda está dentro de los límites del tablero.
   8. ***public void Plantar(int fila, int columna, ArrayList<JPanel> paneles, ArrayList<JLabel> labels):***Método que planta y expande alrededor de una celda vacía en el tablero. Utiliza la recursión para expandir el área vacía.
   9. ***public void handlerExpandir(ArrayList<JPanel> paneles, ArrayList<JLabel> labels, int control):*** Método que maneja la expansión del tablero alrededor de las celdas vacías. Utiliza la recursión para llenar los huecos vacíos en el tablero***.***
   10. ***protected int matrixtoPanel(int fila, int columna, int maxColumna):*** Método protegido que convierte las coordenadas de matriz en un número de panel.
   11. ***private int Colorear(int fila, int columna, ArrayList<JPanel> paneles, ArrayList<JLabel> labels):*** Método privado que colorea una celda en el tablero y muestra el número de minas adyacentes. También se encarga de propagar la expansión si la celda está vacía.

En resumen, la clase "LogicaTablero" contiene la lógica esencial para gestionar un juego de tablero similar al Buscaminas, incluyendo la manipulación de minas, banderitas, conteo de minas adyacentes y la expansión de áreas vacías del tablero.

* **Clase minas**

1. **Responsabilidades:**
   1. La clase "Minas" es responsable de gestionar el estado de una celda, es decir, si la celda contiene una mina o no.
2. **Atributos**
   1. private boolean estado: ***Un atributo booleano que representa si una celda contiene una mina (true) o no (false).***
3. Métodos:
   1. ***public Minas():*** Este es el constructor de la clase "Minas". Se utiliza para inicializar una instancia de "Minas". Por defecto, establece el estado de la celda como false, lo que significa que la celda está vacía y no contiene una mina.
   2. ***public boolean getEstado():*** Método que devuelve el estado actual de la celda. Retorna true si la celda contiene una mina y false si la celda está vacía.
   3. ***public void setEstado(boolean estado):*** Método que permite cambiar el estado de la celda. Puede establecer el estado en true para indicar que la celda contiene una mina o en false para indicar que la celda está vacía.

En resumen, la clase "Minas" es una clase simple que se utiliza para representar el estado de las celdas en el juego de Buscaminas, permitiendo indicar si una celda contiene una mina o no. Los métodos getEstado() y setEstado(boolean estado) permiten obtener y cambiar el estado de una celda, respectivamente.

* **Clase Tablero:**

1. **Responsabilidades:**
   1. La clase "Tablero" es responsable de gestionar el estado y la lógica del tablero de juego, incluyendo la ubicación de las minas y la interacción con la lógica del juego.
2. **Atributos:**
   1. ***private Minas[][] cuadricula:*** Una matriz bidimensional de objetos "Minas" que representa el tablero de juego.
   2. ***private boolean[][] ValidacionMinas:*** Una matriz bidimensional de valores booleanos que se utiliza para realizar validaciones y rastrear el estado de cada celda del tablero.
   3. ***private int cantMinas***: Un entero que almacena la cantidad total de minas en el tablero.
   4. ***private int filas, columnas:*** Enteros que almacenan el número de filas y columnas en el tablero.
   5. ***private LogicaTablero logicaT:*** Una instancia de la clase "LogicaTablero" que se utiliza para gestionar la lógica del tablero.
   6. ***private LogicaMinas logicaM:*** Una instancia de la clase "LogicaMinas" que se utiliza para gestionar la lógica de las minas en el tablero.
3. **Métodos Públicos:**
   1. ***public Tablero():*** Constructor de la clase "Tablero" que inicializa las instancias de las clases "LogicaTablero" y "LogicaMinas".
   2. ***public LogicaTablero getLogicaTablero():*** Método que devuelve la instancia de "LogicaTablero" asociada al tablero.
   3. ***public LogicaMinas getLogicaMinas():*** Método que devuelve la instancia de "LogicaMinas" asociada al tablero.
   4. ***public int getCantMinas():*** Método que devuelve la cantidad total de minas en el tablero.
   5. ***public int getFila()***: Método que devuelve el número de filas en el tablero.
   6. ***public int getColumna():*** Método que devuelve el número de columnas en el tablero.
   7. ***public Minas getCuadricula(int fila, int columna):*** Método que devuelve una celda específica del tablero representada por un objeto "Minas".
   8. ***public boolean getValidacionMinas(int fila, int columna):*** Método que devuelve el estado de validación de una celda específica en la matriz de validación.

* 1. ***public void setValidacionMinas(int fila, int columna, boolean estado):*** Método que establece el estado de validación de una celda específica en la matriz de validación.
  2. ***public void setCantMinas(int cantMinas)***: Método que permite establecer la cantidad total de minas en el tablero.
  3. ***public void setFila(int filas):*** Método que permite establecer el número de filas en el tablero.
  4. ***public void setColumna(int columna):*** Método que permite establecer el número de columnas en el tablero.
  5. ***public void inicializarCuadricula(Tablero T):*** Método que inicializa la cuadrícula del tablero, creando objetos "Minas" para cada celda y estableciendo el estado de validación para cada celda. Luego, llama al método ponerMinas() para colocar las minas en posiciones aleatorias.

1. **Métodos Privados:**
   1. ***private void ponerMinas():*** Método privado que coloca las minas en posiciones aleatorias dentro del tablero.

En resumen, la clase "Tablero" se encarga de gestionar el estado y la lógica del tablero de un juego similar al Buscaminas, incluyendo la ubicación de las minas y la interacción con la lógica del juego a través de las instancias de "LogicaTablero" y "LogicaMinas". También se encarga de inicializar la cuadrícula del tablero y colocar las minas de manera aleatoria.

***Package “presentación”***

* **Clase ControladorInicio**

1. **Responsabilidades:**
   1. La clase "ControladorInicio" se encarga de manejar los eventos del mouse en los componentes de la interfaz gráfica de usuario (GUI) relacionados con el inicio de un juego. Esto incluye responder a los clics del usuario en diferentes botones y campos de texto.
2. **Atributos:**
   1. ***private VistaInicio vI:*** Un atributo que almacena una referencia a la instancia de la clase "VistaInicio", que probablemente es una vista de la interfaz de usuario relacionada con el inicio del juego.
3. **Métodos y funcionalidad:**
   1. ***Constructor: public ControladorInicio(VistaInicio vI):*** El constructor recibe una instancia de "VistaInicio" y la almacena en el atributo vI para que el controlador pueda interactuar con la vista.
   2. Implementación de la interfaz "MouseListener":
   3. Este controlador implementa la interfaz "MouseListener", lo que significa que debe proporcionar una implementación para los siguientes métodos de esa interfaz: mouseClicked, mousePressed, mouseReleased, mouseEntered y mouseExited. En este caso, parece que solo se implementa el método mouseClicked.
   4. ***Método mouseClicked(MouseEvent e):*** Este método es invocado cuando se hace clic en algún componente de la GUI. El método verifica cuál componente generó el evento de clic (usando e.getSource()) y, según el componente, realiza una acción específica:

* Si el clic proviene del primer "botón" (vI.getPN\_1()), se llama al método initTablero del modelo con valores específicos (8 filas, 8 columnas y 10 minas).
* Si el clic proviene del segundo "botón" (vI.getPN\_2()), se llama al método initTablero del modelo con valores diferentes (16 filas, 16 columnas y 40 minas).
* Si el clic proviene del tercer "botón" (vI.getPN\_3()), se llama al método initTablero del modelo con valores aún diferentes (16 filas, 30 columnas y 99 minas).
* Si el clic proviene del cuarto "botón", se recopilan valores de columnas, filas y minas desde campos de texto (vI.getTP\_1(), vI.getTP\_2() y vI.getTP\_3()) y se intenta llamar al método initTablero del modelo con esos valores. Se incluye una verificación de excepción (NumberFormatException) en caso de que el usuario ingrese valores que no sean números.

En resumen, la clase "ControladorInicio" se encarga de manejar los eventos del mouse relacionados con la configuración inicial de un juego. Dependiendo del botón o campo de texto en el que el usuario haga clic, se invocarán diferentes acciones para inicializar el juego con parámetros específicos. También se incluye manejo de excepciones para garantizar que los valores ingresados sean válidos.

* ***Clase ControladorTablero***

1. **Responsabilidades:**
   1. La clase "ControladorTablero" se encarga de manejar eventos del mouse relacionados con la interacción del usuario en un tablero de juego, como hacer clic en casillas, colocar banderas o revelar casillas.
2. **Atributos:**
   1. **private VistaTablero ventanaT:** Un atributo que almacena una referencia a la instancia de la clase "VistaTablero", que probablemente es la vista de la interfaz de usuario relacionada con el tablero de juego.
3. **Métodos y funcionalidad:**
   1. **Constructor: public ControladorTablero(VistaTablero ventanaP)**: El constructor recibe una instancia de "VistaTablero" y la almacena en el atributo ventanaT para que el controlador pueda interactuar con la vista.
   2. Implementación de la interfaz "MouseListener":
   3. Este controlador implementa la interfaz "MouseListener", lo que significa que debe proporcionar una implementación para los siguientes métodos de esa interfaz: mouseClicked, mousePressed, mouseReleased, mouseEntered y mouseExited. En este caso, se implementan los métodos mouseClicked, mouseEntered, mouseExited, y mouseReleased.
   4. **Método mouseClicked(MouseEvent e):** Este método es invocado cuando se hace clic en algún componente de la GUI. En el código proporcionado, este método no contiene ninguna funcionalidad específica, por lo que parece no estar siendo utilizado en este contexto.
   5. **Método mousePressed(MouseEvent e)**: No contiene funcionalidad específica en este contexto y no se utiliza.
   6. **Método mouseReleased(MouseEvent e):** Este método se encarga de manejar los eventos de clic del mouse cuando se libera el botón del mouse. Realiza las siguientes acciones:

* Identifica el componente (en este caso, un panel) que generó el evento de clic utilizando e.getSource().
* Calcula la posición del panel en el arreglo de paneles de juego.
* Verifica si la casilla en la que se hizo clic es una mina o no.
* Si se hace clic en una mina (verMina(fila, columna) devuelve true) con el botón izquierdo del mouse (e.getButton() == 1), se llama al método Explotar para revelar todas las minas y se llama al método Terminar del modelo del juego.
* Si se hace clic en una casilla que no es una mina con el botón izquierdo del mouse, se llama al método Plantar para revelar la casilla y se llama a handlerExpandir para expandir las casillas adyacentes vacías.
* Si se hace clic con el botón derecho del mouse, se llama al método Banderita para colocar una bandera en la casilla.
  1. **Método mouseEntered(MouseEvent e):** Este método se encarga de manejar los eventos cuando el puntero del mouse entra en un componente. En el código proporcionado, cambia el color de fondo del panel si ciertas condiciones se cumplen (por ejemplo, si el panel no tiene una bandera ni ha sido revelado).
  2. **Método mouseExited(MouseEvent e):** Este método se encarga de manejar los eventos cuando el puntero del mouse sale de un componente. En el código proporcionado, restaura el color de fondo del panel si ciertas condiciones se cumplen (por ejemplo, si el panel no tiene una bandera ni ha sido revelado).

En resumen, la clase "ControladorTablero" se encarga de manejar eventos del mouse en un tablero de juego, permitiendo al usuario interactuar con las casillas, revelar minas, colocar banderas y expandir áreas vacías. También controla cambios en el color de fondo de las casillas según la interacción del usuario.

* ***Clase Modelo***

1. **Responsabilidades:**
   1. **Gestión del modelo de juego:** La clase "Modelo" se encarga de gestionar y controlar el modelo de juego, que incluye el tablero y su estado, así como las reglas del juego.
   2. **Control de la interfaz de usuario**: La clase también interactúa con la vista (representada por la clase "Vista") para mostrar información y controlar la interfaz de usuario.
2. **Atributos:**
   1. **private Vista pantalla:** Un atributo que almacena una referencia a la vista de la interfaz de usuario relacionada con el juego.
   2. **private Tablero T:** Un atributo que representa el tablero de juego, que es una instancia de la clase "Tablero" (presumiblemente una representación del modelo del juego).
3. **Métodos y funcionalidad:**
   1. ***Método getTablero():*** Este método devuelve una instancia del tablero del juego. Si el tablero aún no se ha creado, lo crea y lo almacena en el atributo T.
   2. ***Método getPantalla():*** Este método devuelve una instancia de la vista de la interfaz de usuario relacionada con el juego. Si la vista aún no se ha creado, la crea y la almacena en el atributo pantalla.
   3. ***Método initTablero(int filas, int columnas, int cantMinas):*** Este método se utiliza para inicializar el tablero del juego con las dimensiones especificadas y la cantidad de minas. Realiza las siguientes acciones:

* Obtiene una instancia del tablero mediante getTablero().
* Establece las filas, columnas y cantidad de minas en el tablero.
* Inicializa la cuadrícula del tablero llamando al método inicializarCuadricula(T).
* Cierra la pantalla de inicio (si existe) llamando a pantalla.closeInicio().
* Actualiza y redibuja la vista llamando a métodos como pantalla.getTablero() y realizando operaciones de revalidación y repintado.
  1. ***Método iniciar():*** Este método se utiliza para iniciar la vista de la interfaz de usuario relacionada con el juego. Realiza las siguientes acciones:
* Obtiene una instancia de la vista mediante getPantalla().
* Hace que la vista sea visible llamando a pantalla.setVisible(true).
* Muestra la pantalla de inicio llamando a pantalla.getInicio().
  1. ***Método Terminar():*** Este método se utiliza para indicar que el juego ha terminado, posiblemente porque el jugador ha perdido. En la implementación actual, simplemente imprime "Perdiste pa" en la consola. Es posible que esta función debe ser extendida para manejar de manera adecuada el final del juego.

En resumen, la clase "Modelo" es responsable de gestionar el estado y las reglas del juego, interactuar con la vista y controlar la inicialización y finalización del juego. También proporciona métodos para configurar el tablero y mostrar la vista de juego.

* **Vista.java**

1. **Responsabilidades**
   1. Dd
2. **Atributos**
3. **Metodos y funcionalidad**
   1. |dss
   2. }

* **VistaInicio.java**

1. **Responsabilidades**
   1. Dd
2. **Atributos**
3. **Metodos y funcionalidad**
   1. |dss
   2. }

* **VistaInicio.java**

1. **Responsabilidades**
   1. Dd
2. **Atributos**
3. **Metodos y funcionalidad**
   1. |dss
   2. }

* **VistaReiniciarjava**

1. **Responsabilidades**
   1. Dd
2. **Atributos**
3. **Metodos y funcionalidad**
   1. |dss
   2. }

* **VistaTablero.java**

1. **Responsabilidades**
   1. Dd
2. **Atributos**
3. **Metodos y funcionalidad**
   1. |dss
   2. }

**4. Instalación y Configuración**

**4.1 Requisitos de hardware y software:**

Estos son los requisitos que deberá tene tu dispositivo para poder ejecutar el programa con buena eficacia

**Requisitos de Hardware:**

* **CPU:** Cualquier procesador moderno debería ser suficiente. No se requiere una CPU potente para jugar Buscaminas.
* **Memoria RAM:** Una pequeña cantidad de RAM es suficiente para ejecutar el juego de Buscaminas. Incluso 1 GB de RAM es más que suficiente.
* **Tarjeta gráfica:** No se requiere una tarjeta gráfica dedicada. La mayoría de las tarjetas gráficas integradas en las computadoras modernas son adecuadas para el juego.
* **Espacio en disco duro:** El juego de Buscaminas es muy liviano y ocupa una cantidad insignificante de espacio en disco, generalmente menos de 100 MB.
* **Sistema operativo:** El juego de Buscaminas está disponible en una variedad de sistemas operativos, incluyendo Windows, macOS y Linux. Asegúrate de que tu sistema operativo sea compatible con el juego.

**Requisitos de Software:**

* **Sistema operativo:** El juego de Buscaminas suele estar preinstalado en sistemas Windows, por lo que en la mayoría de las computadoras con Windows, no se necesita software adicional. Si deseas jugar en macOS o Linux, es posible que necesites una versión del juego específica para tu sistema operativo o usar una implementación de código abierto.
* **Java (opcional):** Algunas versiones del juego de Buscaminas están escritas en Java. Si estás utilizando una versión basada en Java, necesitarás tener Java Runtime Environment (JRE) instalado en tu computadora.
* **Resolución de pantalla:** La mayoría de las computadoras modernas cumplen con los requisitos de resolución de pantalla para jugar Buscaminas. No se requiere una resolución alta.
* **Sonido (opcional):** Si deseas disfrutar de efectos de sonido en el juego, necesitarás una tarjeta de sonido y altavoces o auriculares conectados.

**4.2 Instrucciones para la instalación**

* Descarga el código fuente: Asegúrate de que tienes la carpeta que contiene el código fuente del juego de Buscaminas en tu computadora. Puede ser un archivo ZIP que descomprime previamente.
* Instala las dependencias (si es necesario): Verifica si el juego requiere la instalación de algún software adicional, como un entorno de desarrollo específico o una versión específica de un lenguaje de programación.
* Compila el código (si es necesario): Si el código fuente no está precompilado, abre una terminal o línea de comandos, navega hasta la carpeta que contiene el código fuente y compila el código utilizando el comando o la herramienta adecuada. Esto puede variar según el lenguaje de programación del juego.
* Ejecuta el juego: Una vez que el código se haya compilado (si es necesario), busca el archivo de inicio o el archivo principal del juego..
* Disfruta del juego: El juego de Buscaminas debería iniciarse y mostrarse en tu pantalla. Juega el juego siguiendo las reglas y los controles proporcionados por el juego.

**5.Uso de la aplicación**

**5.1 Inicio rápido**

* Abrir el juego: Abre el juego Buscaminas en tu sistema operativo. En Windows, puedes encontrarlo en el menú "Inicio" o buscarlo en el cuadro de búsqueda.
* Seleccionar una casilla: Haz clic izquierdo en una casilla para revelarla. El juego comenzará a revelar casillas, y si no hay una mina en la casilla seleccionada, verás un número que indica cuántas minas están adyacentes a esa casilla. Esto te proporciona pistas.
* Identificar minas: Utiliza las pistas numéricas para identificar la ubicación de las minas. Por ejemplo, si una casilla muestra el número "1", significa que una mina está adyacente a esa casilla.
* Marcar minas con banderas: Si crees que una casilla contiene una mina, haz clic derecho en ella para colocar una bandera. Esto marca la casilla como sospechosa de contener una mina.
* Continuar explorando: Continúa seleccionando casillas y marcando minas con banderas hasta que hayas revelado todas las casillas vacías. El juego termina cuando todas las casillas vacías están descubiertas o cuando haces estallar una mina.
* Ganar o perder: Si revelas todas las casillas vacías sin hacer estallar una mina, ganas el juego. Si haces estallar una mina, pierdes y el juego termina

**5.2 Funciones detalladas.**

* Tablero de Juego: El juego se juega en un tablero cuadriculado donde se ocultan casillas que pueden contener minas o números que indican cuántas minas están adyacentes.
* Minas: El objetivo principal es evitar las minas. Si seleccionas una casilla que contiene una mina, el juego termina y pierdes.
* Pistas Numéricas: Las casillas que no contienen minas muestran un número que indica cuántas minas están adyacentes. Estas pistas ayudan a los jugadores a identificar la ubicación de las minas.
* Selección de Casillas: Puedes hacer clic izquierdo en una casilla para revelar su contenido. Si no hay mina en esa casilla, se mostrará un número o estará vacía, revelando casillas adyacentes automáticamente si son seguras.
* Marcación de Minas con Banderas: Puedes hacer clic derecho en una casilla para marcarla con una bandera, indicando que crees que esa casilla contiene una mina. Esto es útil para recordar qué casillas consideras peligrosas.
* Descubrimiento de Casillas Vacías: Cuando seleccionas una casilla vacía (sin número ni mina adyacente), se revelarán automáticamente todas las casillas vacías adyacentes, lo que facilita la progresión en el juego.
* Niveles de Dificultad: El juego generalmente ofrece diferentes niveles de dificultad, que varían en el tamaño del tablero y la cantidad de minas ocultas. Los niveles más difíciles tienen tableros más grandes y más minas.

**6.Conclusiones y referencias**

.