

Desarrollo de Aplicación Sudoku con Java y Swing

Aplicación completa con interfaz gráfica y modo consola

Implementación de algoritmos de generación de tableros

Validación de entradas según reglas oficiales

Desarrollado con Java Swing GUI API

 **por ruben ojeda**

Objetivos del Proyecto



```
(altie; sedan; motor: turbo);
```

```
    calteer((attie).l);  
}  
canals ; susliter");  
}
```

Funcionalidades Principales



Interfaz Java Swing

Componentes visuales interactivos



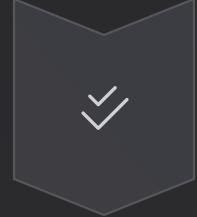
Generación algorítmica

Tableros únicos matemáticamente válidos



Niveles de dificultad

Fácil, medio, difícil



Validación en tiempo real

Retroalimentación inmediata

Arquitectura del Sistema

Modelo
Lógica de generación y validación



Vista
Interfaz gráfica Swing

Controlador
Gestión de interacciones

Implementación Técnica

Algoritmo Recursivo

Generación eficiente de soluciones válidas

Backtracking para resolución de conflictos

Sistema de Validación

Verificación de filas, columnas y cuadrantes

Comprobación instantánea de entradas

Componentes Swing

JFrame para ventana principal

JPanel para celdas del tablero

```
Sudoku solver {
    chidde int lane (antlat io, netimel -vietter hoi, hoi
    / sudin live ();
    sudoku solver);
}

fr surdanu {
    foistarant; vietterian luo,
}
(altie; soden; emiliar taale);

calteer ((lattie).));
}
camnls ; sustiler");
}

engerrition {
    ;; metin tnatention ( );
    medionns, 'täet wettili, dinisine trachin;
    serane. is;, /comker test wort;
    almoile( * );
    suular duale);
}
tadetracking, Lentand osium;
dateertilin, conhuniel uader;
(tanner ualter);
}
conut(); con
```

Interfaz Gráfica



Tablero Principal

Grid 9x9 con celdas interactivas



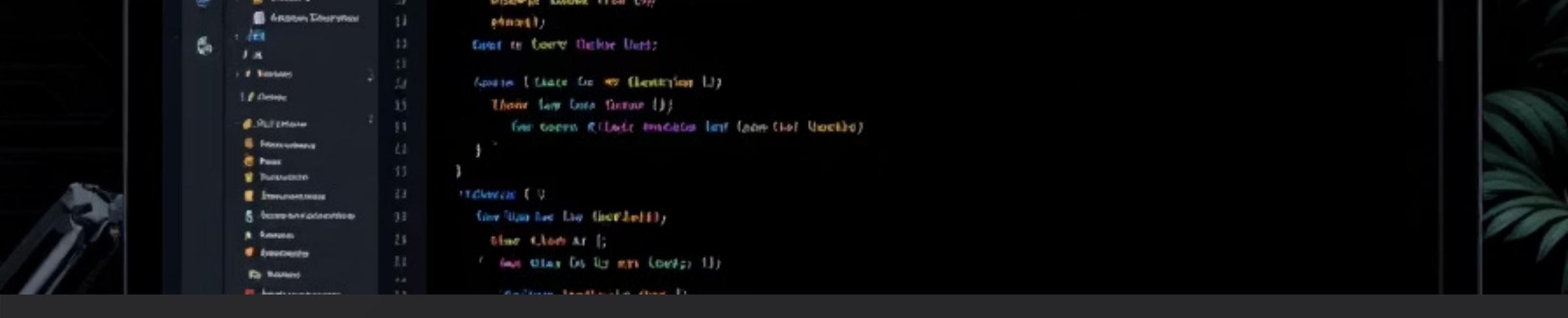
Panel de Control

Botones de funcionalidad intuitivos



Sistema de Validación

Retroalimentación visual por colores



```
private void updateBoard() {
    for (int i = 0; i < board.length; i++) {
        for (int j = 0; j < board[i].length; j++) {
            if (board[i][j] == null) {
                board[i][j] = new Cell();
            }
            board[i][j].update();
        }
    }
}

public void update() {
    for (int i = 0; i < board.length; i++) {
        for (int j = 0; j < board[i].length; j++) {
            if (board[i][j] != null) {
                board[i][j].update();
            }
        }
    }
}

private void checkForWin() {
    int count = 0;
    for (int i = 0; i < board.length; i++) {
        for (int j = 0; j < board[i].length; j++) {
            if (board[i][j] != null) {
                if (board[i][j].isMine() && board[i][j].isFlagged()) {
                    count++;
                } else if (!board[i][j].isMine() && board[i][j].isFlagged()) {
                    count--;
                }
            }
        }
    }
    if (count == 0) {
        System.out.println("¡Ganaste!");
        for (int i = 0; i < board.length; i++) {
            for (int j = 0; j < board[i].length; j++) {
                if (board[i][j] != null) {
                    board[i][j].setMine(true);
                }
            }
        }
    }
}
```

Características Adicionales



Algoritmo de generación mejorado

Usamos la lógica para facilitar la creación de tableros



Sistema de ayuda

Visualización de posibilidades por celda



Doble interfaz

Interfaz tanto en consola como en Java Swing

Conclusiones y Futuro Desarrollo

1

Aplicación Actual

Cumplimiento total de objetivos básicos

2

Próxima Versión

Nuevas funcionalidades y dificultad personalizada

3

Expansión Futura

Rankings + Contador de fallos

4

Versión Avanzada

Algoritmos optimizados de generación (mejores)

