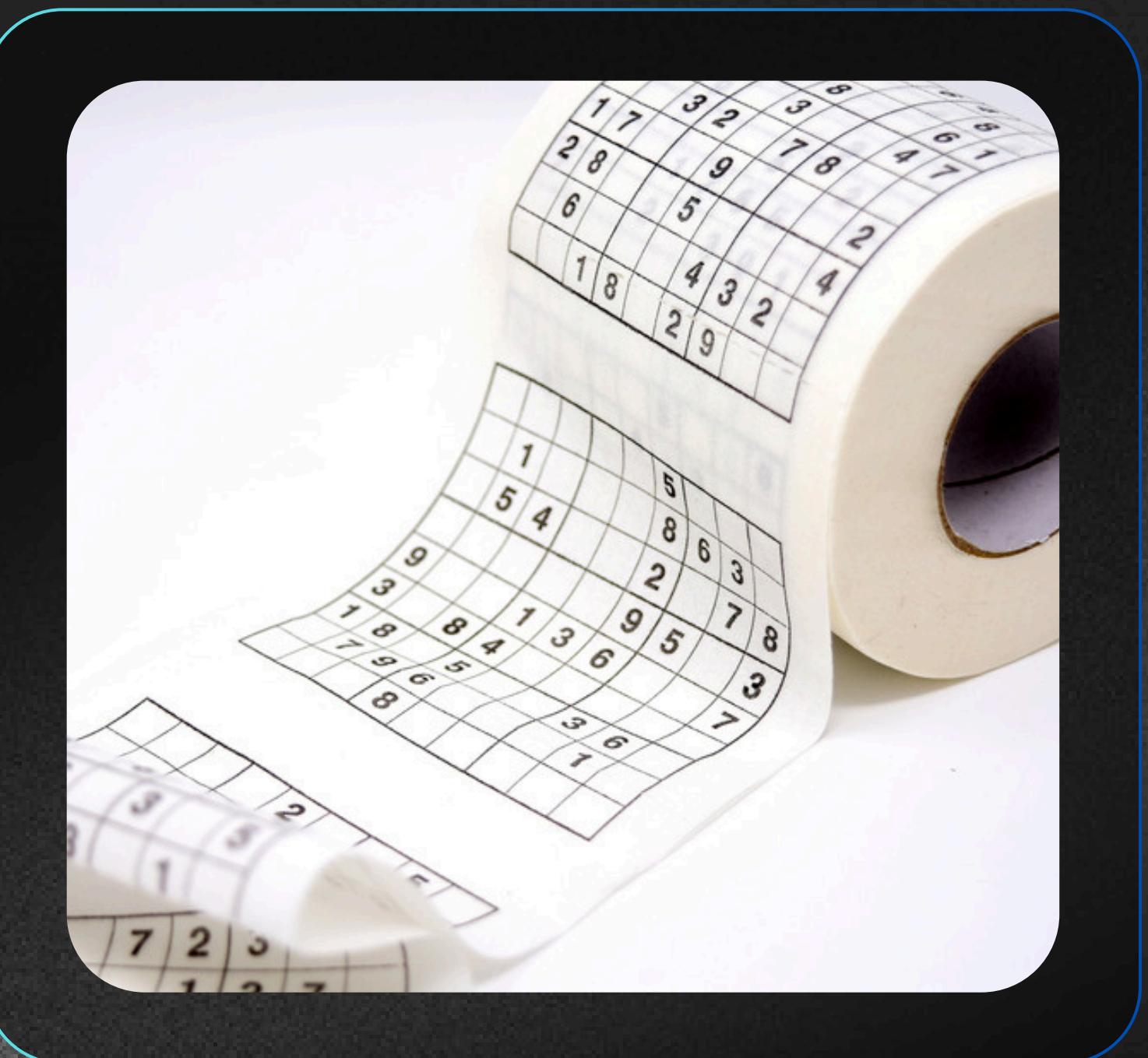


# PROYECTO SUDOKU

Aplicación completa de Sudoku desarrollada en Java, con generación aleatoria de tableros, validación de movimientos, solución automática y una interfaz gráfica interactiva (Swing). Estructurada en paquetes con arquitectura modular, excepciones personalizadas e interfaces para facilitar pruebas y mantenimiento.

Elaborado por: Ismael Vargas Duque



# GENERACIÓN DEL TABLERO

Usamos la clase GeneradorSudoku para crear tableros en tres niveles. Primero llena un tablero completo, luego elimina un número fijo de casillas (35 fácil / 50 medio / 60 difícil) y, antes de entregar el puzzle, comprueba que exista exactamente **UNA** solución única.

**Paquete:** com.ejercicio02.generator

**Clase principal:** GeneradorSudoku

**Interfaz:** IGeneradorSudoku

Fácil

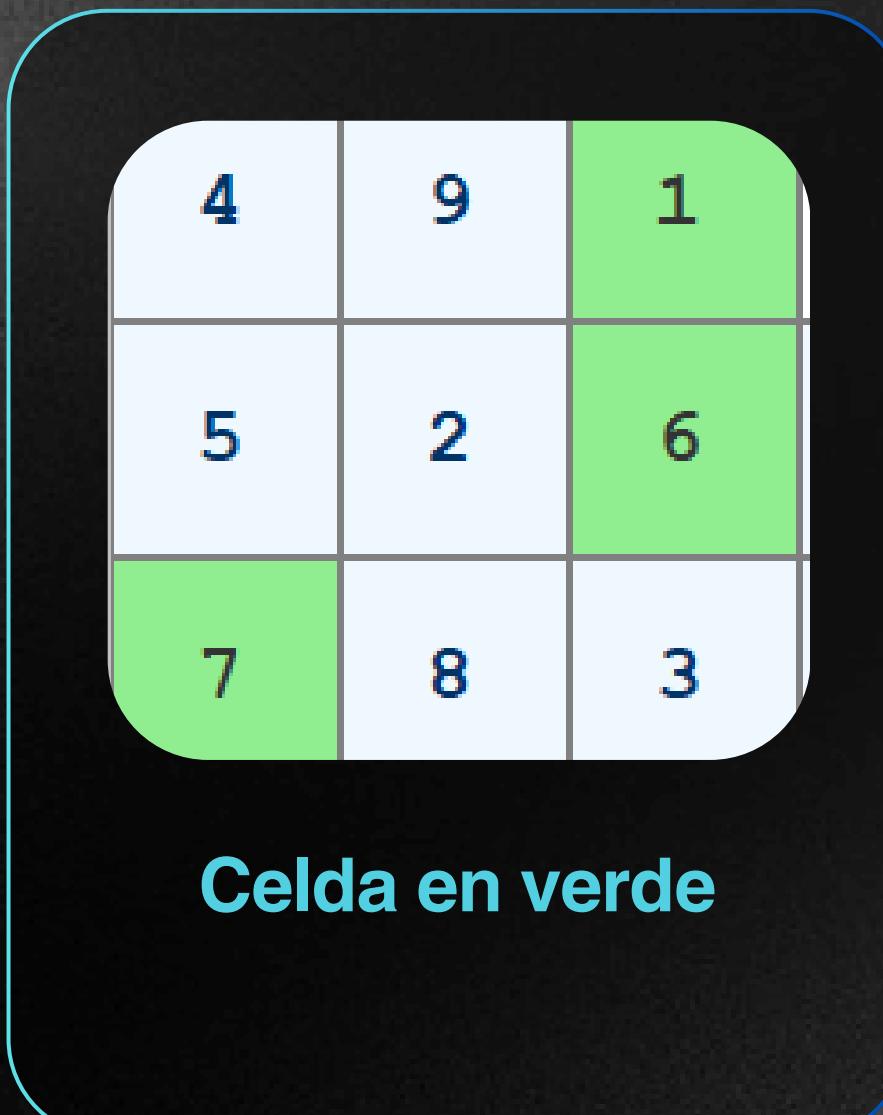
9			7		3	2	1	6
	2	4		6				
5			8	1	7	9	4	
4	9	7		1	3	6	2	
3	1				8			
	5	3		7	9	4		
2	9	6	7	4		3	8	
7				5		2	9	
6	1		4	5	7			

Medio

5			7					5
				3				
1	4			9	3		2	
		5	9		7	4		3
9	7		1			5		
	6	3	2			8	7	
4						3	9	
	9			7	2		6	4

Difícil

1	9							
		7			3			2
			7	1			5	9
			1			4	6	
7	8					4	1	
9				5				3
	1	9			7	6		
4								
			8				3	



## VALIDACIÓN DE MOVIMIENTOS

Cada vez que el usuario introduce un número, el modelo revisa filas, columnas y subcuadrante para detectar conflictos. La GUI marca la celda en rojo si hay error ('ya existe en la fila') o en verde si es válido. Esta validación se puede activar o desactivar desde el menú.

**Paquete:** com.ejercicio02.model

**Clase principal:** Sudoku (método esMovimientoValido)

**GUI:** com.ejercicio02.gui.SudokuGUI (**activación/desactivación, feedback visual**)

# IMPLEMENTACIÓN DEL BACKTRACKING

El algoritmo de backtracking recorre celdas vacías, prueba valores y retrocede automáticamente ante bloqueos. Para optimizar, elige primero la casilla con menos opciones posibles. En pruebas mide menos de 100 ms para resolver cualquier nivel.

**Paquete:** com.ejercicio02.model

**Clase principal:** Sudoku (método recursivo resolverBacktracking)

**Soporte adicional:** com.ejercicio02.game.ResolverSudoku

**Interfaz:** IResolverSudoku

**Recorre el tablero, prueba valores 1–9, retrocede al fallo y copia la solución al terminar.**

**Vuelca el tablero resuelto en solucionGrid.**

**El método resolverSudoku hace una copia del tablero original, llama al backtracking y luego asigna la solución.**

# DISEÑO Y ARQUITECTURA

El proyecto está organizado en seis capas: modelo (reglas), generator, game (consola), gui (Swing), exception y interfaces. Cada módulo comunica a través de interfaces, garantizando separación de responsabilidades y facilidad de mantenimiento.



**Model (lógica y reglas)**



**Generator (creación de puzzles)**



**Game (lógica de consola y Main)**



**Gui (Swing)**



**Exception (excepciones propias)**



**Interfaces (contratos)**

# MANEJO DE EXCEPCIONES

*Hemos definido excepciones específicas:*

**MovimientoInvalidoException**  
**(Jugada ilegal)**

**EntradaFueraDeRangoException**  
**(Valor fuera de 1–9)**

**SudokuException**  
**(Errores genéricos)**

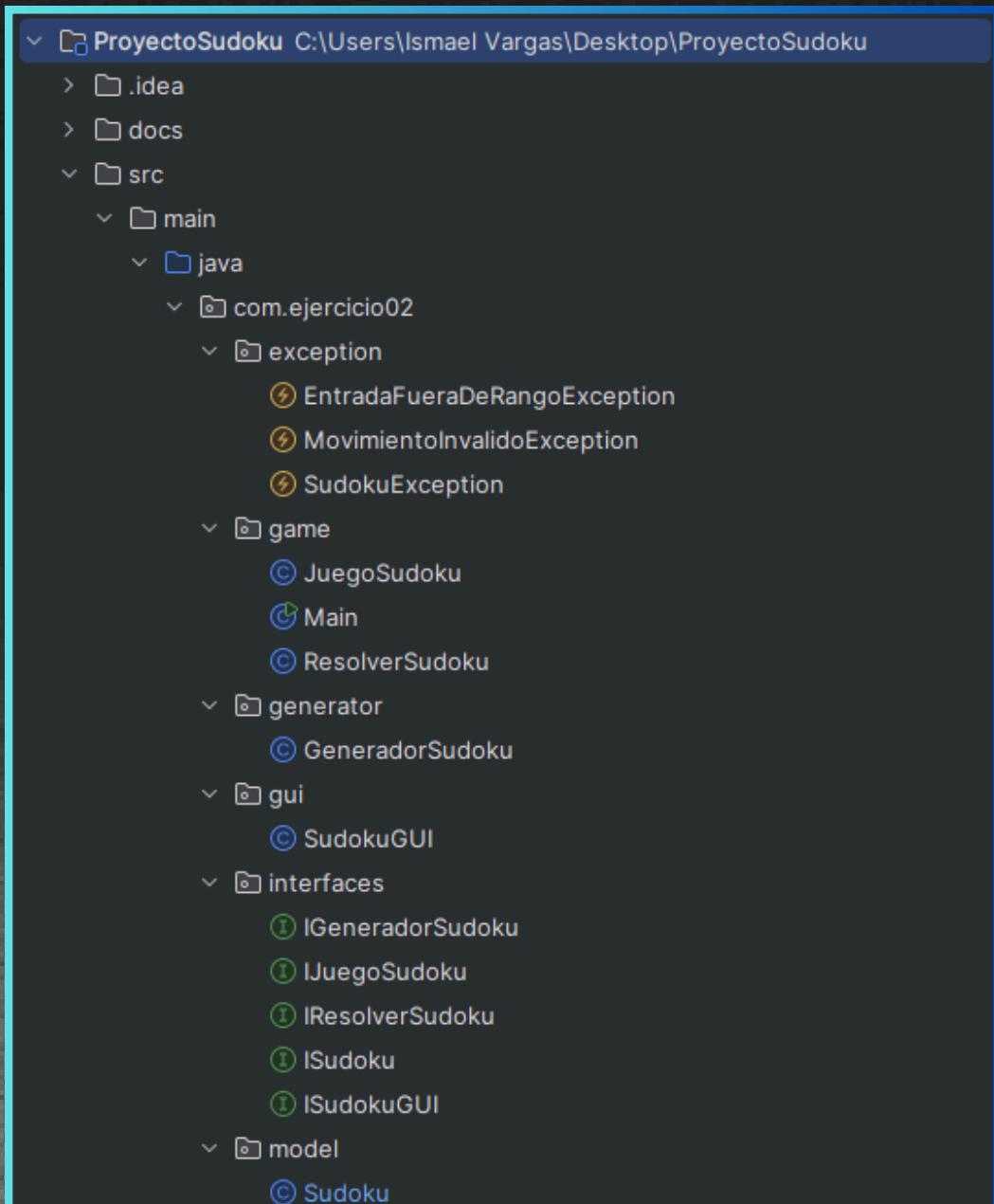
# CALIDAD DEL CÓDIGO Y COMENTARIOS

Nombres expresivos en métodos y clases (p.ej. resolverConBacktracking, getValor).

Paquetes organizados de forma lógica (ver estructura).

Comentarios de alto nivel (Javadoc en interfaces y clases públicas).

Comentarios explican “por qué” y no solo “qué” hace cada método.

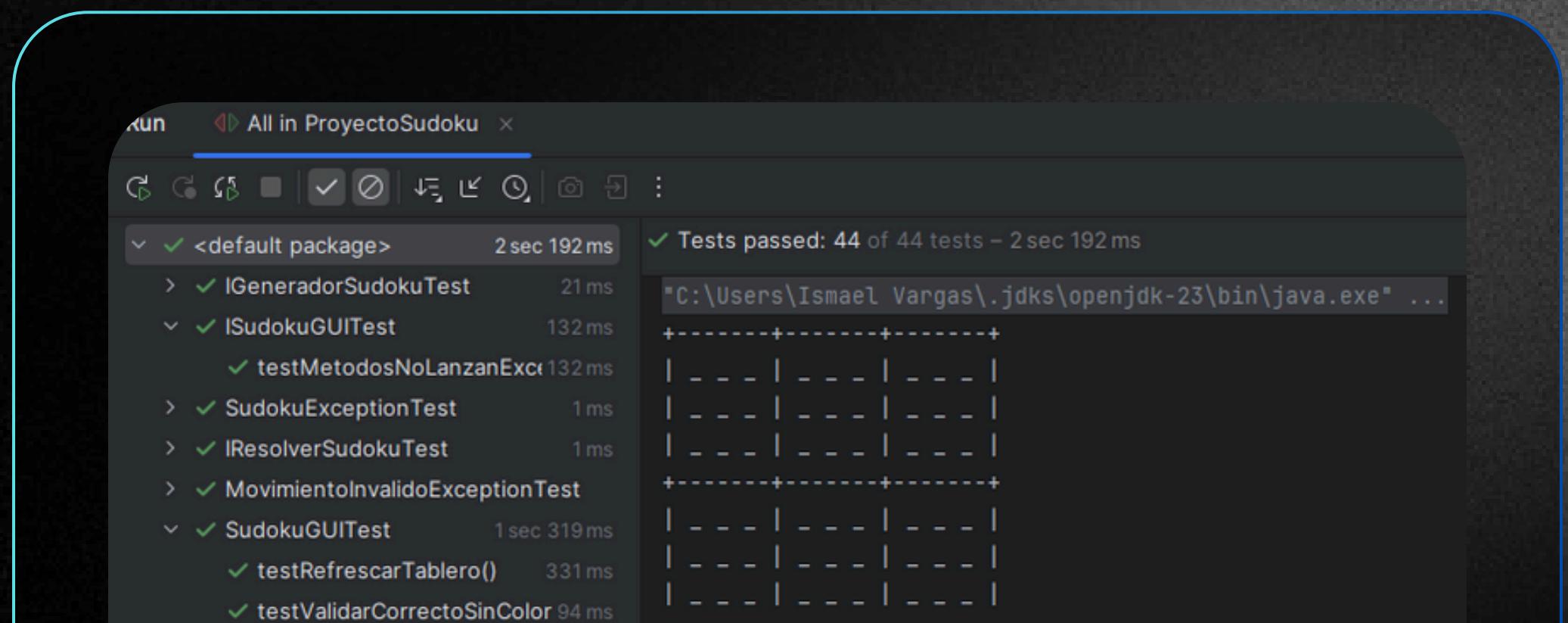


# PRUEBAS UNITARIAS

## Tests destacados:

- SudokuTest (modelo)
- GeneradorSudokuTest
- (mismo paquete game y exception para flujos de error)

Con JUnit 5 cubrimos generación, validación, backtracking y manejo de excepciones. Incluimos tests positivos, negativos y de borde (tableros casi completos). Maven ejecuta toda la suite en cada compilación garantizando calidad constante



The screenshot shows a terminal window with the title "Run All in ProyectoSudoku". The output displays the results of a JUnit 5 test run. It shows 44 tests passed in 2 seconds and 192 milliseconds. The tests are categorized by package and class, with some failing tests indicated by red arrows. The terminal also shows a 9x9 Sudoku grid.

```
JUnit All in ProyectoSudoku
[INFO] Tests passed: 44 of 44 tests – 2 sec 192 ms
[INFO] "C:\Users\Ismael Vargas\.jdks\openjdk-23\bin\java.exe" ...
+-----+
| _ _ _ | _ _ _ | _ _ _ |
| _ _ _ | _ _ _ | _ _ _ |
| _ _ _ | _ _ _ | _ _ _ |
+-----+
| _ _ _ | _ _ _ | _ _ _ |
| _ _ _ | _ _ _ | _ _ _ |
| _ _ _ | _ _ _ | _ _ _ |
+-----+
| _ _ _ | _ _ _ | _ _ _ |
| _ _ _ | _ _ _ | _ _ _ |
| _ _ _ | _ _ _ | _ _ _ |
+-----+
```

Test	Time
<default package> IGeneradorSudokuTest	21 ms
<default package> ISudokuGUITest	132 ms
ISudokuGUITest testMetodosNoLanzanExcepcion	132 ms
<default package> SudokuExceptionTest	1 ms
<default package> IResolverSudokuTest	1 ms
<default package> MovimientoInvalidoExceptionTest	1 ms
<default package> ISudokuGUITest	1 sec 319 ms
ISudokuGUITest testRefrescarTablero()	331 ms
ISudokuGUITest testValidarCorrectoSinColor	94 ms

**GUI Swing:**

SudokuGUI

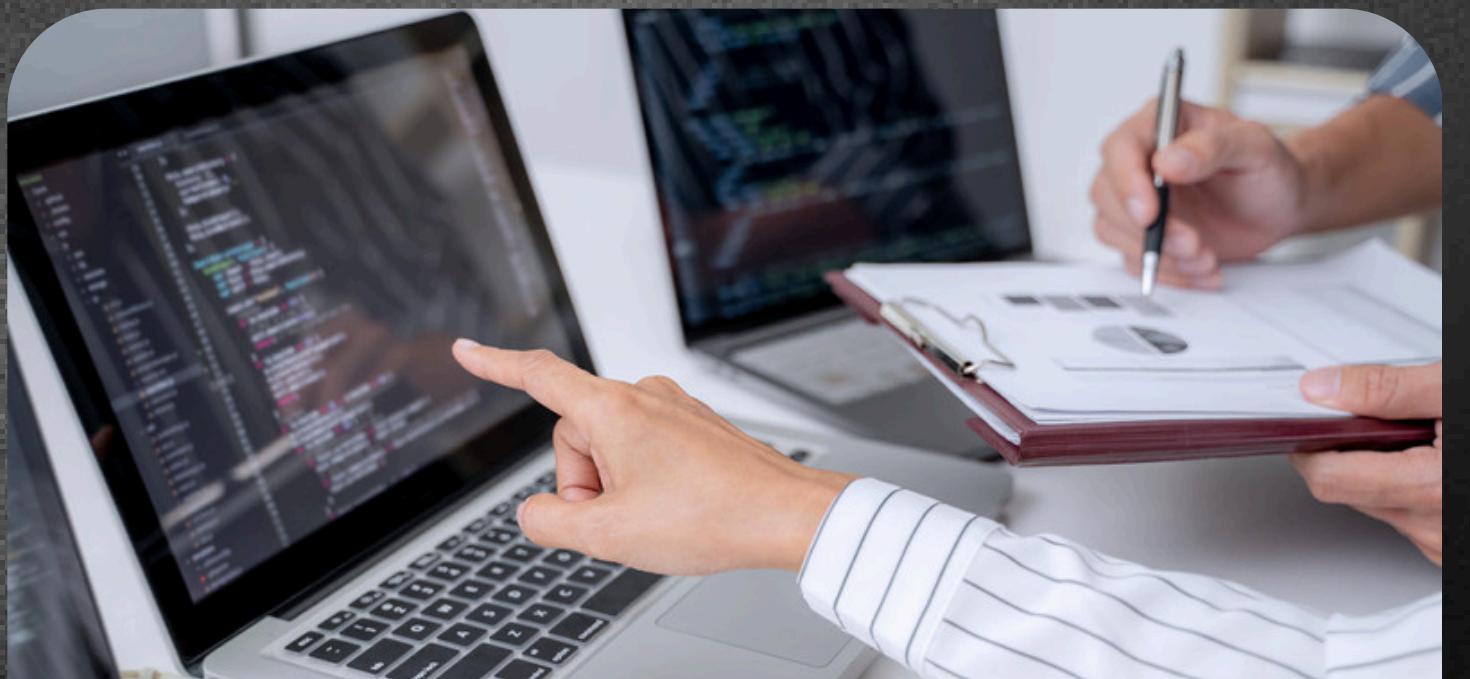
**Consola:**

JuegoSudoku y Main

**Interfaces:**

ISudokuGUI, IJuegoSudoku, etc...

# INTERFAZ DE USUARIO



# DOCUMENTACIÓN

*El repositorio incluye un README con objetivos, análisis, diagramas UML, matriz de trazabilidad e instrucciones de compilación/ejecución (Maven + Main). Además, la carpeta docs contiene todos los diagramas en PDF y ejemplos de trazabilidad para revisión.*

[Documentación Técnica](#)

[UML](#)

[Readme](#)

# ¡GRACIAS!

