Documentación Técnica del Proyecto Sudoku

1. Análisis del Sistema

Este proyecto implementa un juego de Sudoku con componentes siguientes:

1. **Modelo (`com.ejercicio02.model.Sudoku`)**: Gestión del tablero, generación, validación de movimientos y estado de resolución.
2. **Lógica de negocio**:
    - **Generador (`IGeneradorSudoku`, `GeneradorSudoku`)**: Genera tableros según dificultad.
    - **Resolver (`IResolverSudoku`, `ResolverSudoku`)**: Resuelve el Sudoku o proporciona pistas.
    - **Juego en consola (`IJuegoSudoku`, `JuegoSudoku`)**: Flujo de juego interactivo en consola.
3. **Interfaz gráfica (`ISudokuGUI`, `SudokuGUI`)**: GUI Swing para jugar con tablero visual.

Requisitos Funcionales

ID	Descripción
RF1	Generar un tablero de Sudoku según dificultad.
RF2	Validar movimientos (fila, columna, región).
RF3	Colocar números y comprobar estado resuelto.
RF4	Resolver tablero completo y proporcionar pistas.
RF5	Ejecutar juego en consola interactivo.
RF6	Interfaz gráfica con validación en tiempo real.

2. Estrategia de Pruebas

Tipos de casos

- **Positivos**: Operaciones según uso esperado.
- **Negativos**: Inputs inválidos o situaciones erróneas.
- **Borde**: Valores en extremos de rangos (1–9, casillas límite, tableros vacíos/completos).

3. Matriz de Trazabilidad

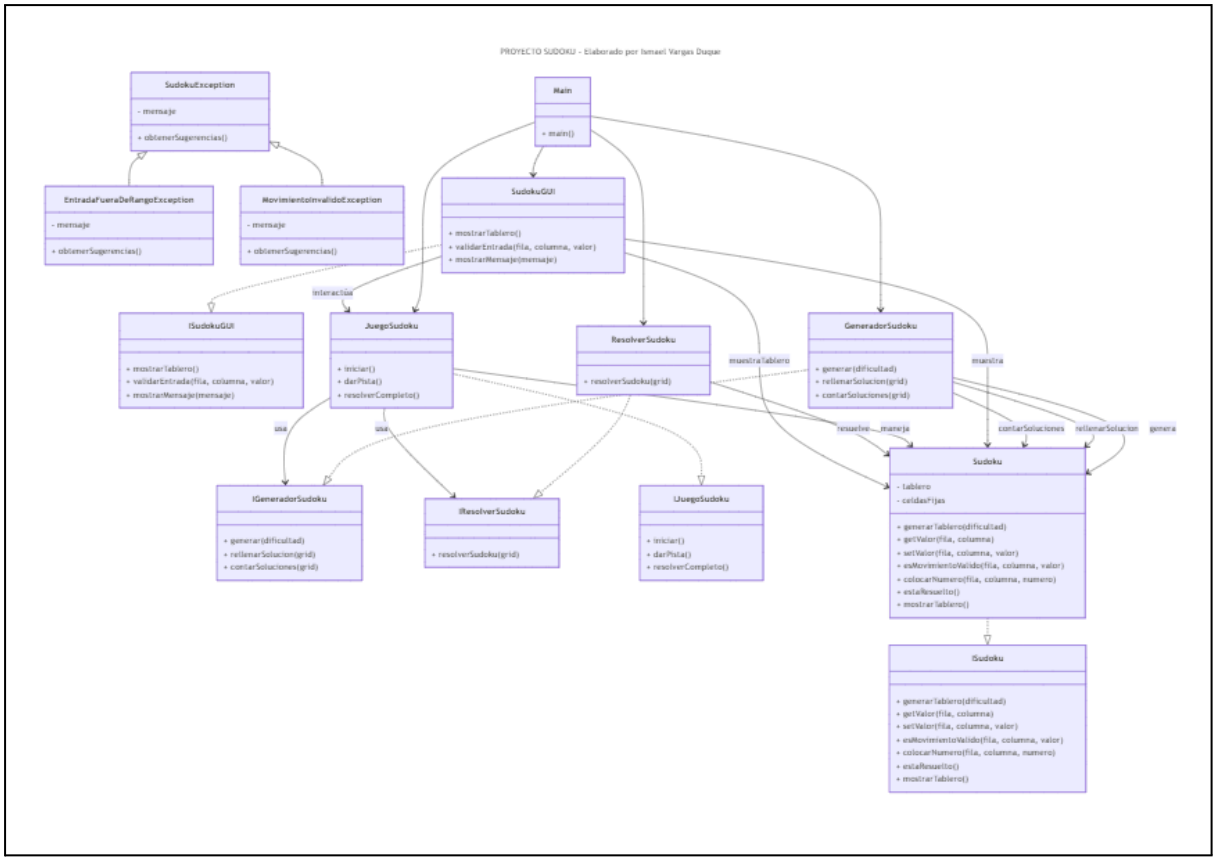
Relación entre RF (Requisitos Funcionales) y casos de prueba.

RF	Clase / Método	Caso de Prueba ID	Tipo	Descripción breve
RF1	**GeneradorSudoku.generar(dificultad)**	TC1_Pos	Positivo	Generar tablero fácil y comprobar 9×9
RF1	**GeneradorSudoku.generar(dificultad)**	TC2_Neg	Negativo	Dificultad inválida -> excepción o fallback
RF1	**GeneradorSudoku.generar(dificultad)**	TC3_Borde	Borde	Límite de dificultad ("Hard")
RF2	**Sudoku.esMovimientoValido**	TC4_Pos	Positivo	Movimiento válido en celda vacía
RF2	**Sudoku.esMovimientoValido**	TC5_Neg	Negativo	Movimiento fuera de rango o duplicado
RF2	**Sudoku.esMovimientoValido**	TC6_Borde	Borde	Casilla (0,0) y (8,8) extremos
RF3	**Sudoku.colocarNumero**	TC7_Pos	Positivo	Colocar número válido y verificar getValor
RF3	**Sudoku.colocarNumero**	TC8_Neg	Negativo	Colocar número inválido -> excepción
RF3	**Sudoku.estaResuelto**	TC9_Pos	Positivo	Tablero completo -> true

RF3	**Sudoku.estaResuelto**	TC10_Borde	Borde	Tablero casi completo (una celda vacía)
RF4	**ResolverSudoku.resolver**	TC11_Pos	Positivo	Resolver sudoku válido completo
RF4	**ResolverSudoku.resolver**	TC12_Neg	Negativo	Sudoku sin solución -> excepción
RF4	**ResolverSudoku.obtenerPista**	TC13_Pos	Positivo	Obtener primera pista en tablero parcial
RF4	**ResolverSudoku.obtenerPista**	TC14_Neg	Negativo	Tablero sin huecos -> excepción
RF5	**JuegoSudoku.iniciar**	TC15_Pos	Positivo	Flujo básico con entradas válidas
RF5	**JuegoSudoku.iniciar**	TC16_Neg	Negativo	Entrada de fila/columna no numérica
RF5	**JuegoSudoku.iniciar**	TC17_Borde	Borde	Finalizar justo en completar última casilla
RF6	**SudokuGUI.validar**	TC18_Pos	Positivo	Entrada correcta resalta verde
RF6	**SudokuGUI.validar**	TC19_Neg	Negativo	Entrada incorrecta resalta rojo
RF6	**SudokuGUI.validar**	TC20_Borde	Borde	Entrada vacía (borrar valor) -> fondo blanco

4. Diagrama UML

PREVIEW UML



4.- Elaborado por Ismael Vargas