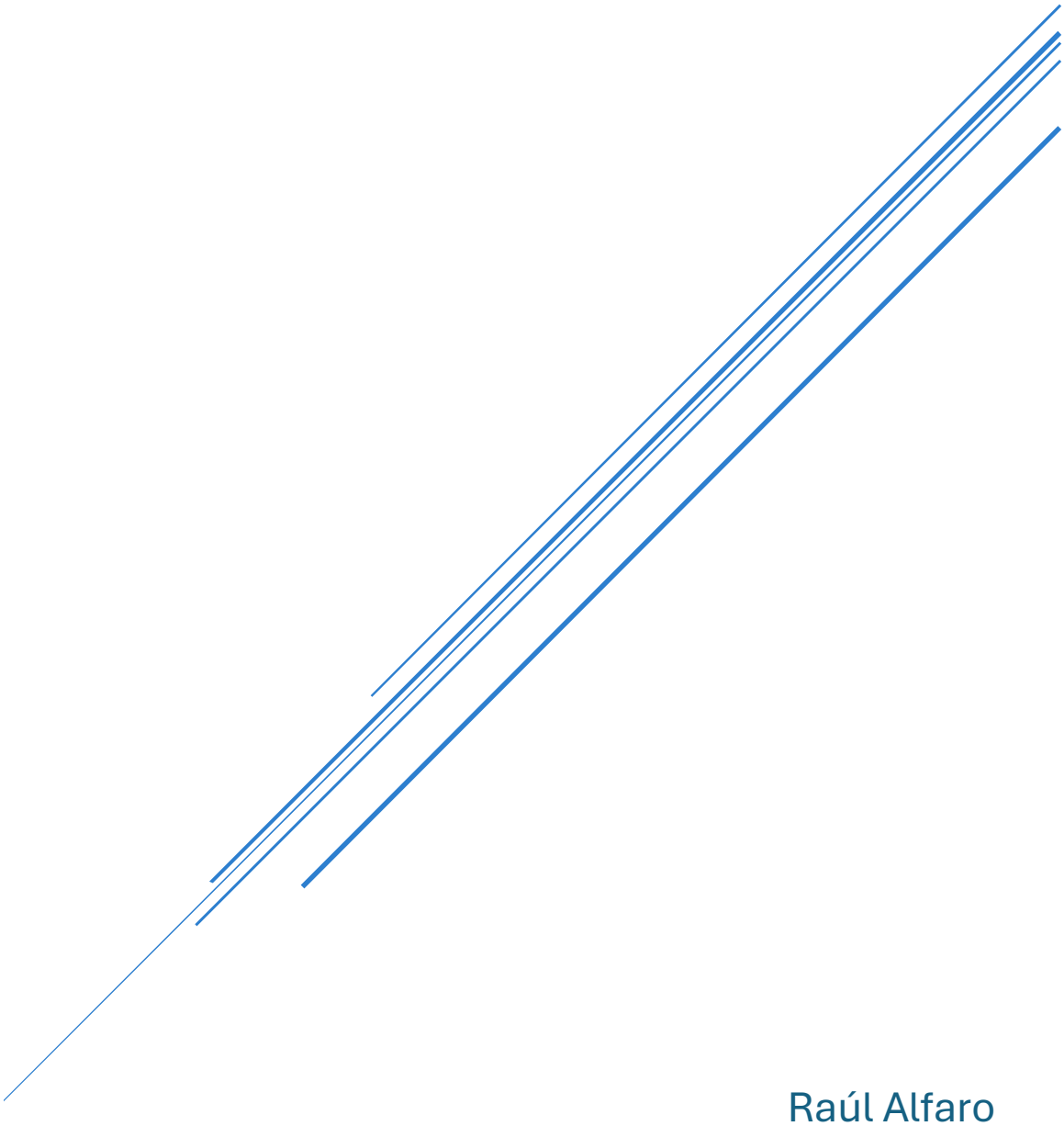
DOCUMENTO DE REQUERIMIENTOS

Sudoku



Raúl Alfaro
Eyden Su Díaz

DESCRIPCIÓN GENERAL

Objetivo: Desarrollar un sistema interactivo de Sudoku utilizando programación lógica (Prolog) como núcleo, que permita:

- Generar tableros válidos y aleatorios con solución única (17-25 pistas iniciales).
- Ofrecer funcionalidades básicas de juego (verificación, sugerencias, resolución automática).
- Integrar opcionalmente una interfaz gráfica (Java, C# o Web) para mejorar la experiencia de usuario.

Alcance:

1. **Lógica del Juego**
    - Generación y validación de tableros 9x9 en Prolog.
    - Resolución automática basada en restricciones lógicas.
    - Cumplimiento estricto de las reglas del Sudoku (sin repeticiones en filas, columnas o regiones 3x3).
2. **Interacción Básica**
    - Interfaz web para ingresar números y acceder a funciones clave:
        - Nuevo juego, Reiniciar, Verificar, Sugerencia, Ver solución.
    - Visualización del tablero y feedback al usuario (errores, celdas vacías).

Limitaciones

- La generación de tableros 100% aleatorios con solución única y la lógica del juego debe generarse completamente en Prolog.
- No se requiere inteligencia artificial avanzada para sugerencias (solución aleatoria o simple es suficiente).
- Priorizar el cumplimiento de los requisitos obligatorios antes de implementar extras.

LISTA DE REQUERIMIENTOS

Requerimientos de Usuario

Funcionales

ID: RU-F001

Descripción: Los usuarios deben poder visualizar un tablero de Sudoku 9x9 con celdas editables y pistas iniciales (17-25 números).

Importancia: Alta

Estimación Inicial: 2 días

Estrategia de Implementación: Representar el tablero como una matriz en Prolog y mostrarlo en interfaz web

ID: RU-F002

Descripción: Los usuarios deben poder ingresar números en celdas vacías mediante coordenadas (X, Y) o clics (UI gráfica).

Importancia: Alta

Estimación Inicial: 1 día

Estrategia de Implementación: Validar coordenadas y actualizar la matriz en Prolog.

ID: RU-F003

Descripción: Los usuarios deben solicitar una sugerencia aleatoria (máximo 5 por partida).

Importancia: Media

Estimación Inicial: 1 día

Estrategia de Implementación: Usar predicados de Prolog para seleccionar una celda vacía y llenarla con un valor válido.

ID: RU-F004

Descripción: Los usuarios deben poder verificar el estado actual del tablero, recibiendo feedback de errores y celdas vacías.

Importancia: Alta

Estimación Inicial: 1 día

Estrategia de Implementación: Validar filas/columnas/regiones con restricciones lógicas en Prolog.

ID: RU-F005

Descripción: Los usuarios deben poder reiniciar el tablero al estado inicial o generar un nuevo juego.

Importancia: Media

Estimación Inicial: 1 día

Estrategia de Implementación: Resetear la matriz o invocar el generador de tableros.

No Funcionales

ID: RU-NF001

Descripción: El sistema debe garantizar que los tableros generados tengan solución única (mínimo 17 pistas).

Importancia: Alta

Estimación Inicial: 3 días

Estrategia de Implementación: Implementar un algoritmo de generación con backtracking y validación de unicidad.

ID: RU-NF002

Descripción: La interfaz debe ser intuitiva (consola o gráfica) con instrucciones claras.

Importancia: Media

Estimación Inicial: 2 días

Estrategia de Implementación: Diseñar menús simples o UI básica con etiquetas descriptivas.

Requerimientos de Sistema

Funcionales

ID: RS-F001

Descripción: El sistema debe generar tableros aleatorios 9x9 cumpliendo las reglas del Sudoku.

Importancia: Alta

Estimación Inicial: 3 días

Estrategia de Implementación: Usar CLP(FD) en Prolog para restricciones y aleatoriedad.

ID: RS-F002

Descripción: El sistema debe resolver el tablero automáticamente al solicitar "Ver solución".

Importancia: Alta

Estimación Inicial: 2 días

Estrategia de Implementación: Implementar un solver basado en backtracking en Prolog.

ID: RS-F003

Descripción: El sistema debe registrar estadísticas por partida (errores, sugerencias usadas, etc.).

Importancia: Media

Estimación Inicial: 1 día

Estrategia de Implementación: Usar variables globales o estructuras en Prolog para almacenar métricas.

No Funcionales

ID: RS-NF001

Descripción: El sistema debe responder a las acciones del usuario en el menor tiempo posible.

Importancia: Media

Estimación Inicial: 1 día

Estrategia de Implementación: Optimizar predicados en Prolog y evitar recursividad infinita.

ID: RS-NF002

Descripción: El sistema debe ser compatible con SWI-Prolog y opcionalmente con Java/C#/Web.

Importancia: Alta

Estimación Inicial: 2 días

Estrategia de Implementación: Documentar dependencias.

Requerimientos de Software

Funcionales

ID: RSO-F001

Descripción: El software debe implementar las reglas del Sudoku (sin repeticiones en filas/columnas/regiones).

Importancia: Alta

Estimación Inicial: 2 días

Estrategia de Implementación: Definir restricciones lógicas en Prolog usando all_different/1.

ID: RSO-F002

Descripción: El software debe permitir la integración con una UI externa (Java/C#/Web) mediante APIs o librerías.

Importancia: Media

Estimación Inicial: 3 días

Estrategia de Implementación: Usar JPL (Java) o HTTP server (Web) para comunicación.

No Funcionales

ID: RSO-NF001

Descripción: El código en Prolog debe ser modular y documentado (comentarios y README en GitHub).

Importancia: Alta

Estimación Inicial: 1 día

Estrategia de Implementación: Dividir en módulos (generación, solución, UI) y usar estándares de documentación.

ID: RSO-NF002

Descripción: El software debe usar control de versiones (GitHub) con commits descriptivos.

Importancia: Alta

Estimación Inicial: 1 día

Estrategia de Implementación: Crear un repositorio público en Github.