Entrega Proyecto #1 - Mancala

Adrian Eduardo Ruiz Cerquera
Leonardo Velázquez Colin
Gustavo Ortiz Loaiza
Tomas Henrique Martinez Gomez

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
ANALISIS DE ALGORITMOS
BOGOTÁ, D.C.
2025

Descripción general del Sistema

El sistema es una aplicación web interactiva del juego Mancala, permitiendo partidas entre dos jugadores humanos (próximamente también jugadores sintéticos) de manera local a través de una interfaz web. El juego sigue las reglas tradicionales del Mancala, gestionando el flujo de turnos, la distribución de semillas, y la determinación del ganador.

El sistema está dividido en dos partes: un frontend Angular que proporciona la interfaz de usuario, y un backend desarrollado con FastAPI que se encarga de la lógica del juego, manejo de partidas y comunicación mediante endpoints HTTP.

Objetivo del documento

El objetivo de este documento es definir la estructura del sistema Mancala, incluyendo sus componentes, comportamiento, flujos de navegación, e interacción entre cliente y servidor. Este documento sirve como referencia para desarrolladores, testers y partes interesadas, permitiendo una comprensión clara del funcionamiento e implementación del sistema.

Requerimientos del sistema

- R01. El sistema debe permitir que un usuario pueda jugar contra la computadora (el sistema).
    - Prioridad: Alta
    - Justificación: Es necesario para tener la disponibilidad de jugar en solitario.
    - Medible: Cuando el sistema pueda jugar contra el usuario, y este tenga la capacidad de determinar un ganador.
    - Tipo de requisito (FURPS+): Funcional
- R02. El sistema debe permitir que un usuario (Jugador 1) pueda jugar contra otro usuario (Jugador 2).
    - Prioridad: Alta
    - Justificación: Es necesario para tener la disponibilidad de jugar contra otro jugador.
    - Medible: Cuando un usuario pueda jugar contra otro usuario, y el sistema tenga la capacidad de determinar un ganador.
    - Tipo de requisito (FURPS+): Funcional
- R03. El sistema debe permitir que la computadora pueda jugar contra la computadora (computadora vs computadora).
    - Prioridad: Alta
    - Justificación: Es útil para realizar pruebas y análisis del juego sin intervención humana.

    - Medible: Cuando el sistema juegue automáticamente ambos turnos hasta llegar al final del juego.
    - Tipo de requisito (FURPS+): Funcional
- R04. El sistema debe contar con un tablero de Mancala con sus respectivos agujeros (6 por cada jugador) y casas bien representados (2 por cada jugador).
    - Prioridad: Alta
    - Justificación: Es necesario para que los jugadores puedan visualizar el estado del juego.
    - Medible: Cuando el sistema muestre el tablero de juego correctamente con los elementos visibles.
    - Tipo de requisito (FURPS+): Funcional
- R05. El sistema debe tener un indicador que comunique de quién es el turno.
    - Prioridad: Alta
    - Justificación: Permite que los jugadores sepan cuándo deben jugar.
    - Medible: Cuando el sistema cambie el indicador de turno correctamente después de cada movimiento.
    - Tipo de requisito (FURPS+): Funcional
- R06. El sistema debe tener un botón para reiniciar la partida.
    - Prioridad: Media
    - Justificación: Permite comenzar una nueva partida sin reiniciar la aplicación.
    - Medible: Cuando el sistema reinicie correctamente el estado del juego al presionar el botón.
    - Tipo de requisito (FURPS+): Funcional
- R07. El sistema debe permitir que los jugadores seleccionen un agujero para mover las habas (piedras).
    - Prioridad: Alta
    - Justificación: Es fundamental para la interacción del usuario con el juego.
    - Medible: Cuando el sistema detecte correctamente la selección de un agujero y mueva las habas en consecuencia.
    - Tipo de requisito (FURPS+): Funcional
- R08. El sistema debe distribuir correctamente las habas (piedras) en sentido antihorario.
    - Prioridad: Alta
    - Justificación: Es necesario para seguir las reglas del juego.
    - Medible: Cuando el sistema realice la distribución en la dirección correcta en cada turno.
    - Tipo de requisito (FURPS+): Funcional
- R09. El sistema debe implementar la regla de captura.
    - Prioridad: Alta
    - Justificación: Es necesario para cumplir con las reglas del Mancala.

    - Medible: Cuando el sistema capture correctamente las habas del oponente al cumplirse la condición de captura.
    - Tipo de requisito (FURPS+): Funcional
- R10. El sistema debe detectar las condiciones para que se dé el final del juego.
    - Prioridad: Alta
    - Justificación: Es necesario para que la partida concluya correctamente.
    - Medible: Cuando el sistema muestre el mensaje de finalización y uno de los lados esté vacío.
    - Tipo de requisito (FURPS+): Funcional
- R11. El sistema debe contar con un diseño intuitivo y accesible.
    - Prioridad: Baja
    - Justificación: Mejora la experiencia de usuario.
    - Medible: Cuando el sistema utilice colores adecuados, tamaños de fuente legibles y controles intuitivos.
    - Tipo de requisito (FURPS+): Usabilidad
- R12. El sistema debe usar algoritmos optimizados para la distribución de semillas y validación de reglas.
    - Prioridad: Media
    - Justificación: Garantiza un rendimiento eficiente.
    - Medible: Cuando el tiempo de ejecución no supere los 100ms en dispositivos de gama media.
    - Tipo de requisito (FURPS+): Performance (desempeño)
- R13. El sistema debe dar un turno extra al jugador si la última semilla cae en su almacén (casa).
    - Prioridad: Alta
    - Justificación: Es una regla clave del juego.
    - Medible: Cuando el sistema otorgue automáticamente un turno adicional en dicha condición.
    - Tipo de requisito (FURPS+): Funcional
- R14. El sistema solo debe permitir al jugador mover las semillas de sus propios agujeros.
    - Prioridad: Alta
    - Justificación: Evita movimientos inválidos.
    - Medible: Cuando el sistema no permita seleccionar agujeros del oponente.
    - Tipo de requisito (FURPS+): Funcional
- R15. El sistema debe determinar al ganador al finalizar el juego.
    - Prioridad: Alta
    - Justificación: Permite cerrar la partida y declarar un ganador de forma justa.
    - Medible: Cuando se muestra el mensaje del ganador y este concuerde con quien tiene más semillas en su almacén.

- Tipo de requisito (FURPS+): Funcional

Alcance de la interfaz y del sistema

Interfaz de Usuario (Frontend):

- Pantalla de inicio donde el usuario puede comenzar una nueva partida.
- Formulario de ingreso de nombres de los dos jugadores.
- Tablero interactivo de Mancala:
  - Muestra el estado actual de los pozos y almacenes (pits y mancalas).
  - Permite a los jugadores seleccionar pozos para realizar movimientos.
  - Actualiza automáticamente el tablero después de cada jugada.
  - Indica el turno actual.
  - Muestra quién ganó al finalizar la partida.

Backend:

- Desarrollado en FastAPI y expone endpoints RESTful para:
  - Iniciar una partida.
  - Realizar movimientos.
  - Obtener el estado actual del juego.
  - Reiniciar el juego.
- Contiene la lógica de negocio para:
  - Validación de jugadas.
  - Reglas del juego (capturas, turnos extra, fin del juego).
  - Determinación del ganador.

Tecnologias Utilizadas

Frontend (Cliente):

- Angular: Framework moderno para construir aplicaciones SPA.
- TypeScript: Lenguaje con tipado fuerte utilizado en Angular.
- Angular Router: Para la navegación entre vistas (inicio, formulario, tablero).

Backend (Servidor):

- FastAPI: Framework web de alto rendimiento en Python para crear APIs REST.
- Uvicorn: Servidor ASGI para ejecutar la aplicación FastAPI.
- Python: Para validación y serialización de datos.

Diagramas

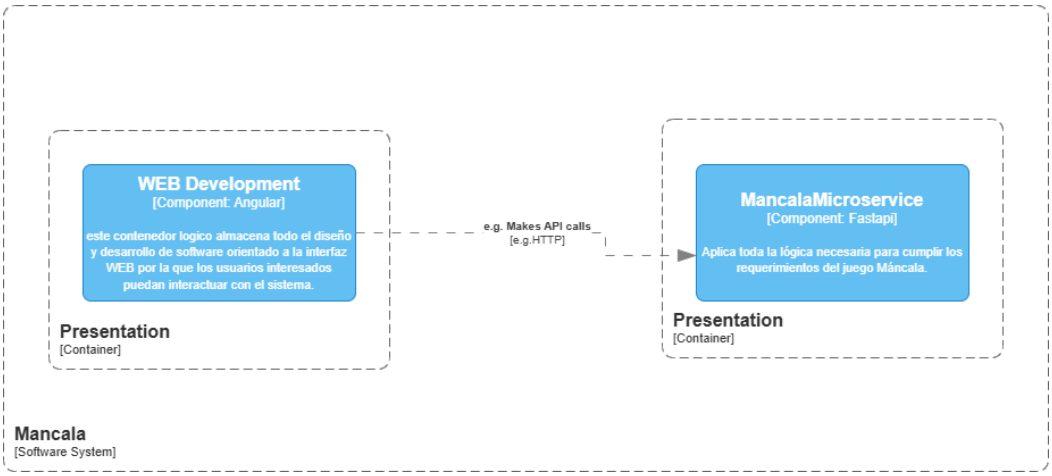

Ilustración 1. Diagrama de Componentes

Este sistema se divide en dos contenedores principales dentro de la capa de presentación.

El primer contenedor es WEB Development, un componente desarrollado en Angular, cuya función es el manejo de la interfaz web. A través de esta interfaz los usuarios pueden interactuar con el sistema de forma intuitiva y accesible.

El segundo contenedor es MancalaMicroservice, un componente implementado con FastAPI, responsable de ejecutar toda la lógica necesaria para satisfacer los requerimientos del juego Mancala. Ambos componentes se comunican mediante llamadas a API, permitiendo así una separación clara entre la lógica de presentación y la lógica del negocio.

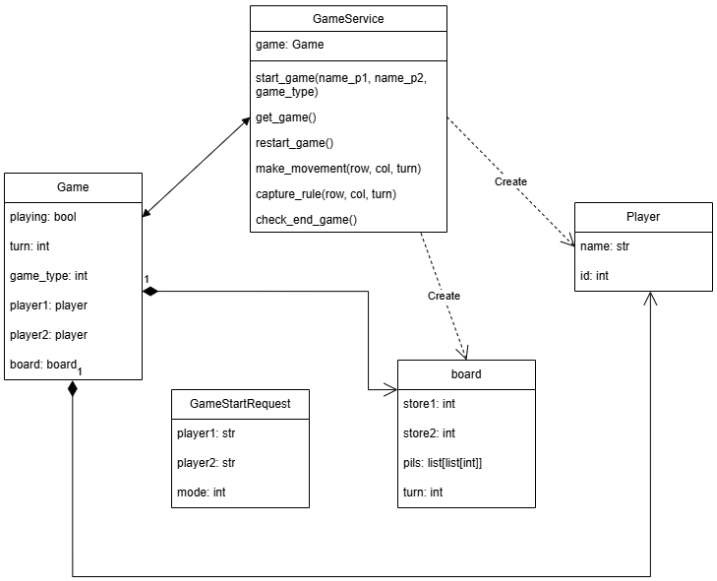

Ilustración 2. Diagrama de Clases

El diagrama representa la arquitectura orientada a objetos de un juego Mancala.

La clase principal es GameService, que controla el flujo del juego: iniciar (start_game), obtener estado (get_game), reiniciar, mover fichas, aplicar reglas y verificar el fin del juego. Esta clase contiene una instancia de Game, que representa el estado del juego.

Game almacena la información clave: jugadores (player1, player2), tablero (board), turno, tipo de juego y si está activo.

Board modela el tablero, con las casas de cada jugador, el turno y la distribución de fichas (pils).

GameStartRequest actúa como DTO para la comunicación entre frontend y backend, incluyendo los nombres de los jugadores y el modo de juego.

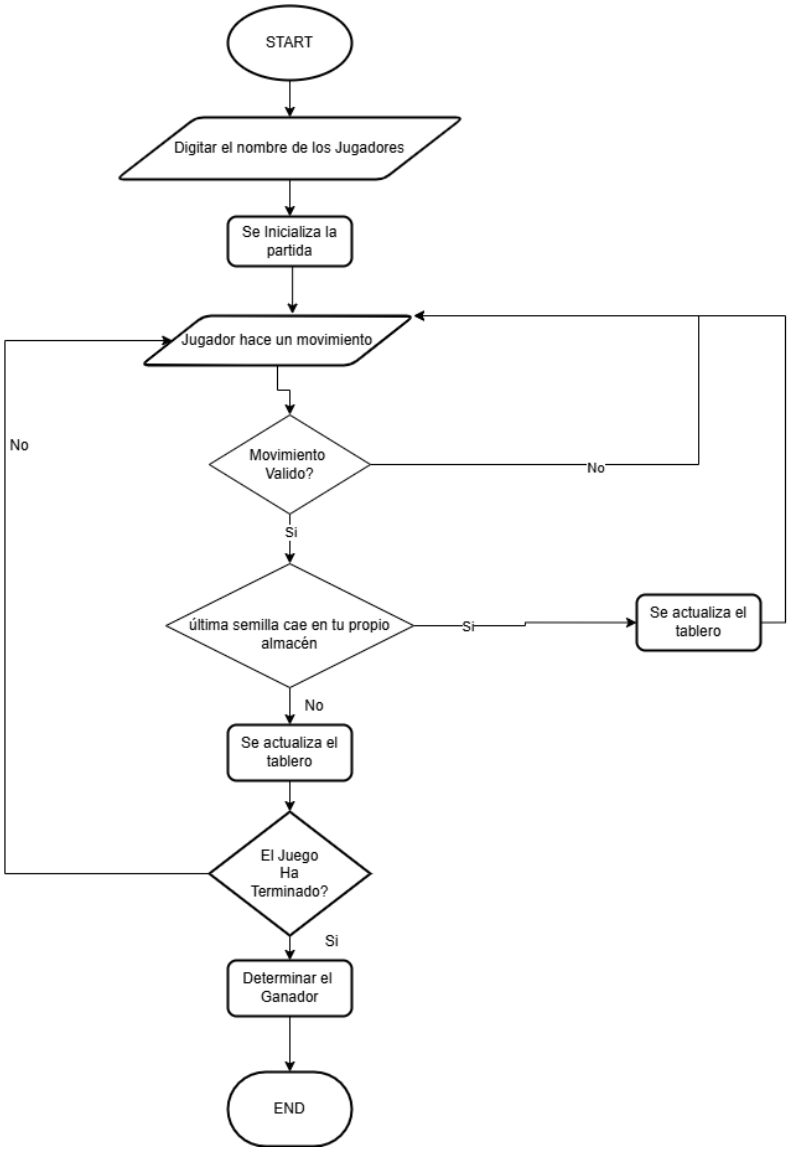

Ilustración 3. Diagrama de Flujo

El diagrama de flujo representa el ciclo básico del proyecto del juego Mancala. El proceso inicia cuando los jugadores ingresan sus nombres y se inicializa la partida. Luego, los jugadores realizan movimientos por turnos. Cada movimiento es validado por el sistema; si no es válido, se solicita un nuevo intento. Si el movimiento es válido, se verifica si la última semilla cae en el propio almacén del jugador; de ser así, este conserva el turno. En cualquier caso, el tablero se actualiza tras cada jugada válida, aplicando las reglas del juego de Mancala. A continuación, se evalúa si el juego ha llegado a su fin. Si aún no ha terminado, continúa el ciclo de turnos; si ha finalizado, se determina el ganador y concluye el juego.