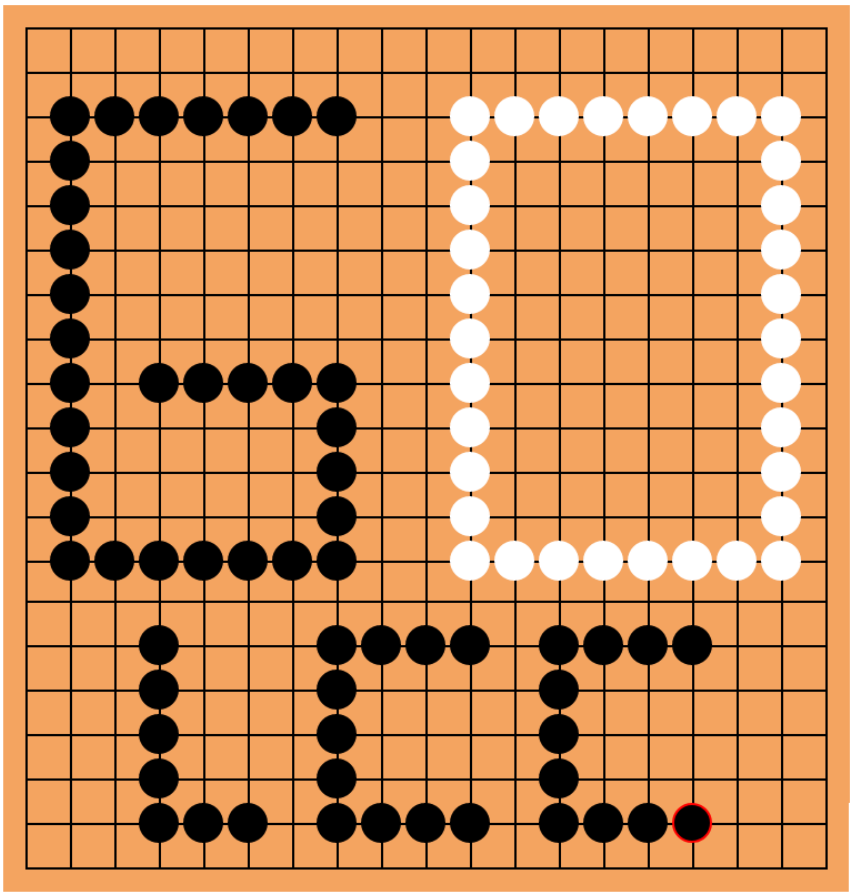
# Lógica para Ciencias de la Computación

## Proyecto N°1




**Autores:**

Carlos Loza – LU: 94399

Javier Fernández Tierno – LU: 106243

# Índice

# Descripción general del software

## Introducción

El GO es un juego de mesa de estrategia para dos personas que consiste de un tablero de 19 líneas horizontales (filas) por 19 líneas verticales (columnas) y piezas llamadas piedras (fichas) de color blanco y negro. Se originó en China hace más de 2500 años. El objetivo del juego es controlar una cantidad de territorio mayor a la del oponente. Para controlar un área, debe rodearse con las piedras. Gana el jugador que controla la mayor cantidad de territorio al finalizar la partida. El juego consiste en colocar las piedras en las intersecciones del tablero. Las negras inician la partida y una vez colocada una piedra, no se puede volver a mover. Se puede capturar una piedra o un conjunto de piedras y eliminarlas del tablero si están completamente rodeadas por piedras de otro color.

### Inicio de la partida

Al inicio de la partida el tablero está vacío. Las negras juegan primero. Después ambos jugadores juegan alternadamente. Una jugada consiste en poner una piedra en una intersección vacía. En general, el jugador más fuerte juega con blancas, puesto que las negras tienen la ventaja de iniciar primero.

### Captura

Cuando un jugador hace una jugada que priva de su última libertad a una piedra del oponente, debe sacar las piedras rodeadas del tablero y guardarlas separadamente hasta el final de la partida. Las piedras que solo poseen una libertad se dicen que están en atari.

### Suicidio

No se permite hacer una jugada ocupando una posición libre en el interior de una formación enemiga, a no ser que esta jugada capture piedras enemigas. Si la formación que uno quiere capturar está rodeada, entonces se puede jugar en su interior, si al hacerlo gana dicha formación.

### Pasar el turno

En lugar de poner una piedra, un jugador puede pasar, es decir pierde un turno. Pasar un turno casi nunca es buena idea. Solo es bueno pasar de turno al final de la partida, cuando los territorios ya están definidos y no hay jugadas por hacer. Cuando los dos jugadores pasan consecutivamente, se acaba la partida.

### Final de la partida

Una vez finalizada la partida, se retiran las piedras muertas de cada bando añadiéndolas a las capturadas. Se conocen como piedras muertas a aquellas que se encuentran rodeadas y en grupos con mala forma. Entonces, los jugadores cuentan los puntos de cada territorio, teniendo en cuenta los espacios encerrados por las fichas de cada jugador y se decide quién ganó la partida.

# Descripción de la implementación

Se utilizó el lenguaje de programación PROLOG para llevar a cabo la implementación lógica del videojuego. También se utilizaron tres lenguajes básicos de desarrollo web: HTML, CSS y JavaScript.

La estrategia principal de este videojuego es detectar, luego de colocar una ficha, si las fichas adyacentes quedan en situación de encierro, si estas fichas a su vez tienen fichas adyacentes que también cumplen la característica de encierro, luego de visitadas, estas serán eliminadas del tablero.

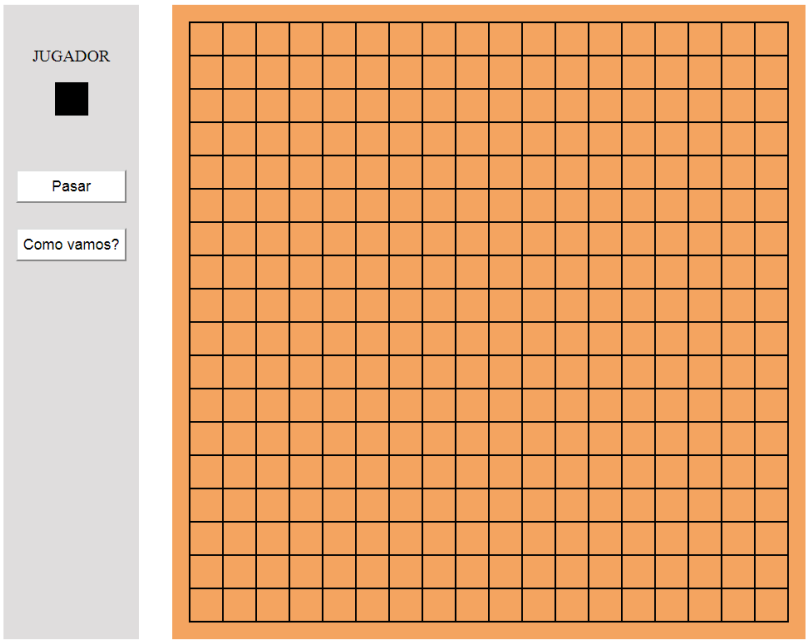
Como funcionalidad requerida, además de detectar fichas encerradas, el videojuego detecta también si las fichas colocadas quedan en situación de suicidio, invalidando esta jugada. Además de la situación de suicidio otra jugada inválida es colocar una ficha donde ya existe otra.

El videojuego finalizara luego de que ambos jugadores pasen su turno simultáneamente, se mostrara el resultado final de la partida y a posterior se generara un nuevo tablero vacío, comenzando así una nueva partida.

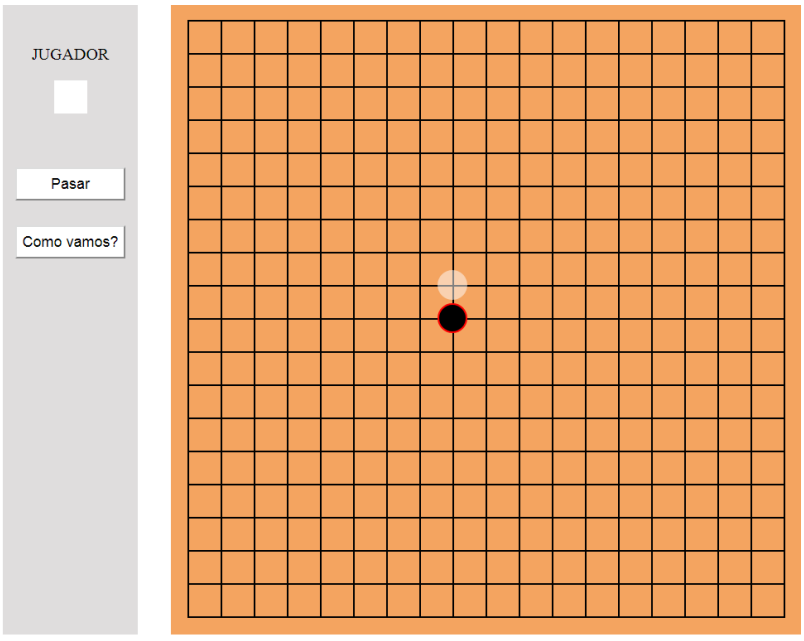
Algunas reglas del juego de mesa físico fueron omitidas en esta implementación.

# Documentación para el cliente

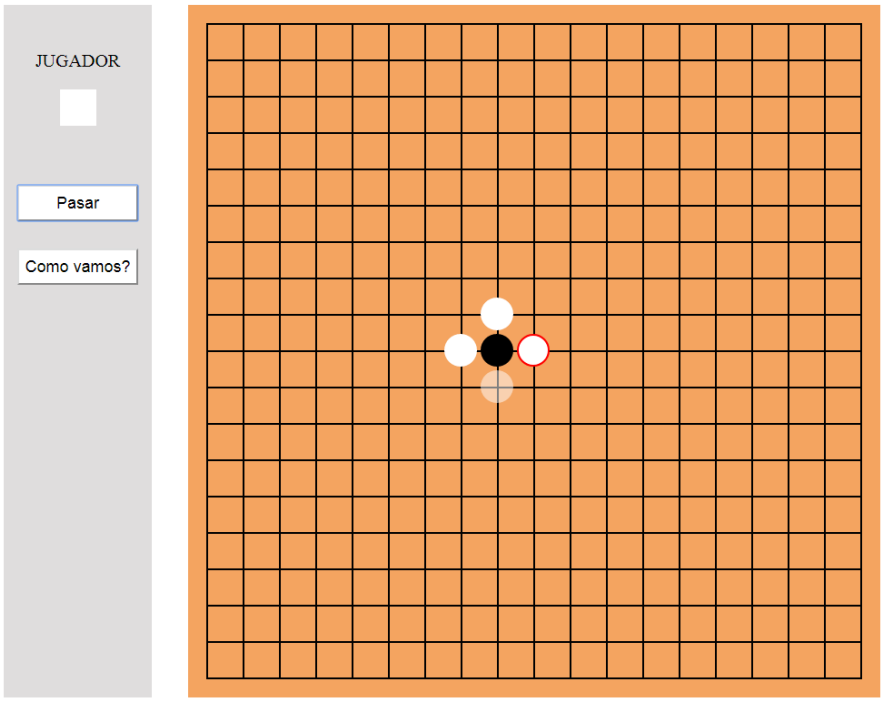
El juego consta de una grilla de 19x19 vacía al iniciar la partida. También se cuenta con los botones “Pasar” que permite pasar de turno y “Como vamos?” que permite conocer los puntos de la partida en el momento que se desee. Por último, siempre empieza el jugador con fichas negras.



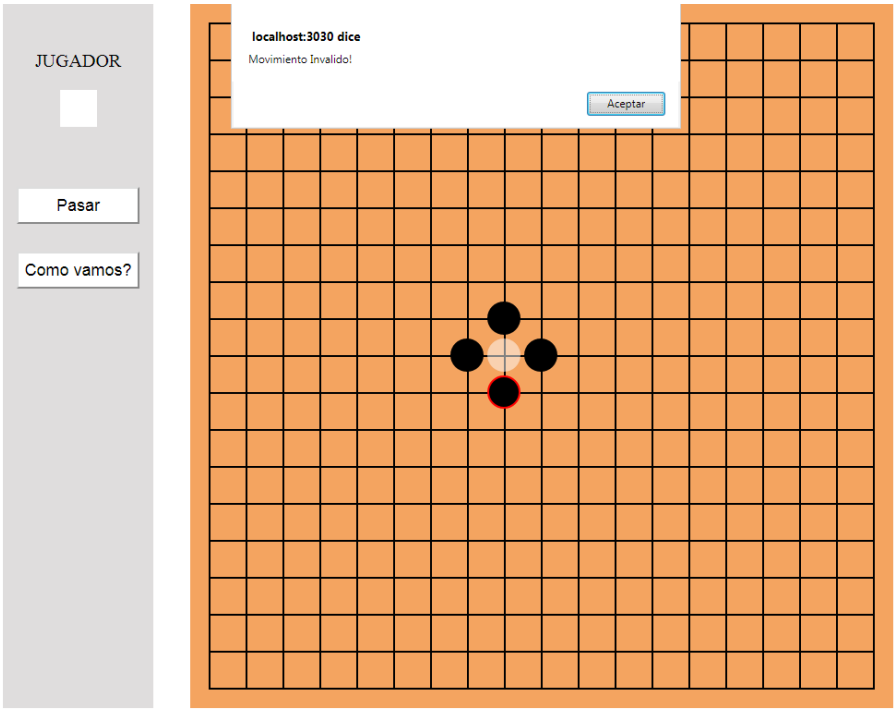

Luego de colocarse una ficha, sigue el turno del jugador con fichas del color opuesto. Es decisión de este jugar una ficha o pasar su turno.



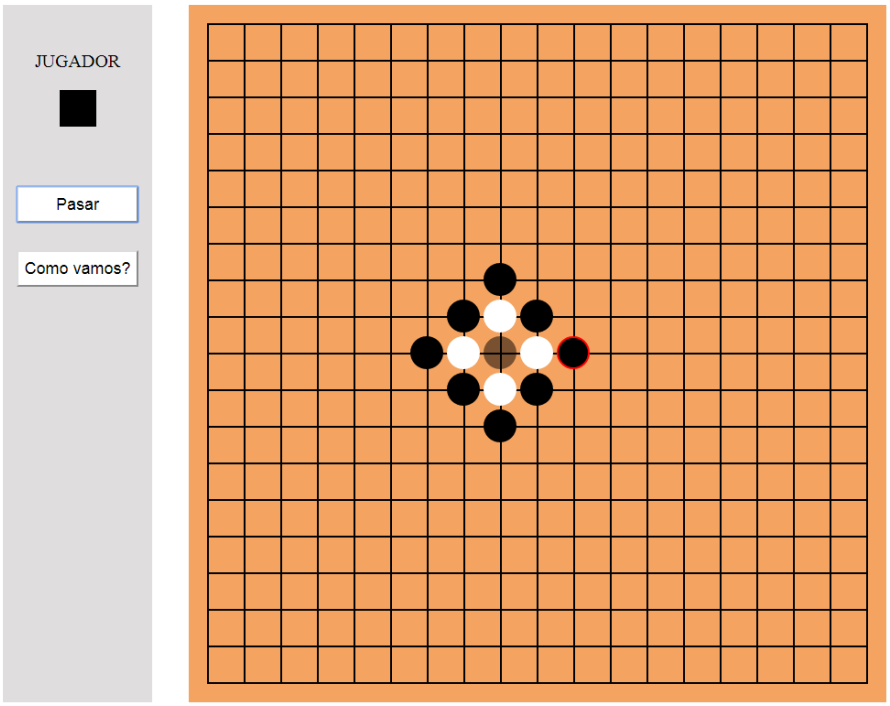
A continuación, se presenta una situación llamada “atari” que consta de una ficha negra siendo encerrada por fichas blancas, cuando el jugador coloque una ficha blanca en la parte inferior de la ficha negra, esta quedara eliminada y retirada del juego.



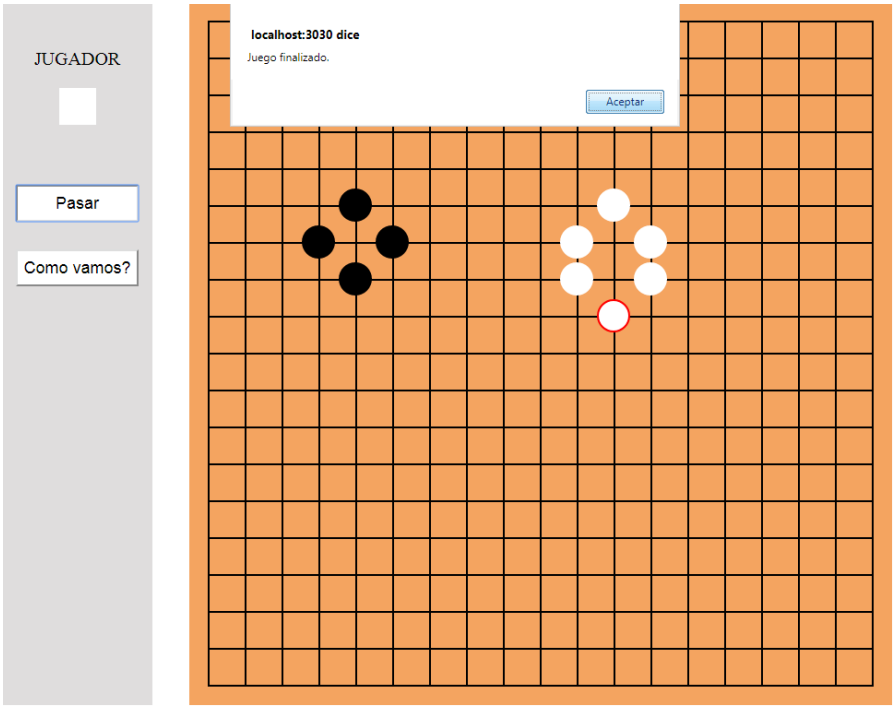

La siguiente situación es considerada suicidio, ya que el jugador con fichas blancas intenta colocar una ficha entre medio de fichas negras, quedando así encerrada automáticamente. Esto es ilegal para el videojuego y no se permite colocar fichas en este caso y similares.



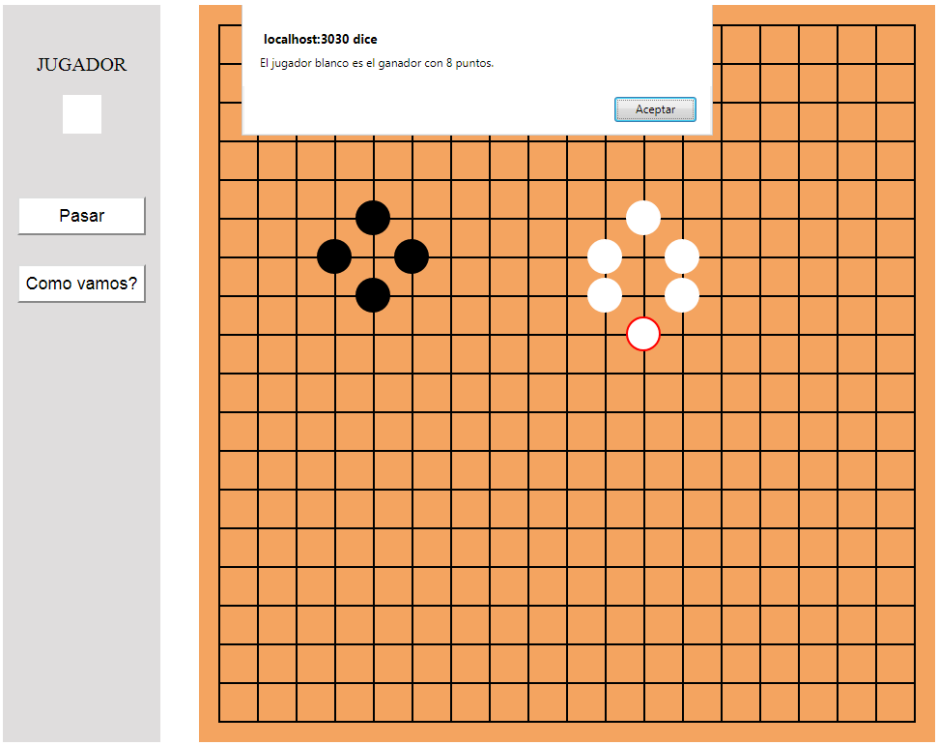
Sin embargo, la siguiente jugada no es considerada suicidio, ya que el jugador con fichas negras puede colocar una ficha en el centro de esta formación, quedando así eliminadas las fichas de su adversario, ya que quedan encerradas por fichas negras. Estas fichas blancas son eliminadas y retiradas de la partida.



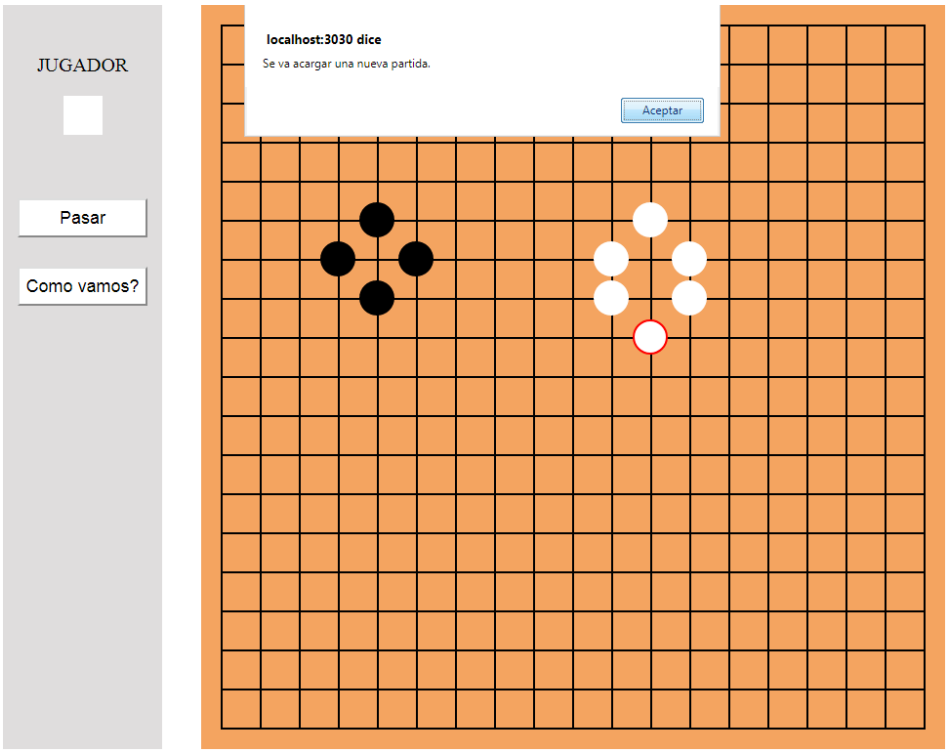

Cuando ambos jugadores decidieron pasar su turno, consecutivamente el juego finalizara y procederá a calcular los puntos de cada jugador, mostrando al ganador.



A continuación inmediata se muestra al ganador de la partida y su correspondiente puntaje.




Luego de finalizada la partida, se procederá a crear una nueva partida.



# Documentación para el desarrollador

A continuación se describen detalles acerca de funciones implementadas, relaciones entre funciones y relaciones entre interfaces.

## Implementación en Prolog

Predicados que provee la cátedra:

- **emptyBoard(Board):** Provee un tablero formado por una lista que contiene listas como elementos.
- **goMove(Board, Color, [Fila,Col], RRBoard):** Dado un tablero Board, el color Color y una posición [Fila, Columna], se retorna el tablero RRBoard con una nueva ficha colocada en la posición especificada.
- **replace(X, XIndex, Y, [Xi|Xs], [Xi|XsY]):** Este predicado tiene sentido su uso en dos etapas, para el caso de esta aplicación web se aplica dicho predicado en primer lugar para obtener la fila buscada luego con el segundo replace se modifica esta fila en la posición de la columna deseada y se inserta esta fila modificada en el tablero.

Predicados creados por los autores:

- **suicidio(Board, Fila, Col, Color):** Dado un tablero, verifica si en la posición Fila y Columna, la ficha de color Color queda en situación de suicidio.
- **reemplazarBoard(Ant ,Board, F, C, Color, RBoard):** Utiliza el replace que provee la cátedra para reemplazar una ficha colo Ant por una ficha color Color en la posición [F,C] del tablero Board y retorna el tablero modificado en RBoard.
- **limpiarAlrededor(Board, Fila, Col, Color, RBoard):** Dado un tablero y una posición [Fila, Columna] verifica si la ficha de color Color encierra a sus adyacentes. Si las encierra devuelve un tablero sin esas fichas encerradas, caso contrario, devuelve el tablero sin modificaciones.
- **invertirColor(Color, ColorI):** Dado un color Color devuelve el color opuesto en ColorI.
- **cascaraLimpiarEncerrado(Board, Fila, Col, Color, Board):** Es un cascara de encerrado que sirve para devolver un tablero modificado si hubo encierro o, caso contrario, el mismo tablero sin modificaciones.
- **encerrado(Board, Fila, Col, ColorE, ColorV, ColorR, RBoard):** Dado un tablero y una posición [Fila, Columna] verifica si es una ficha de color ColorV (Victima) encerrada por fichas color ColorE (Encerrador) y reemplaza esta ficha por una ficha de color ColorR (Reemplazo). Se considera como límite de encierre fichas color ColorE y también los bordes del tablero.

- **contarFichas(Board, CantBlancas, CantNegras):** Es un predicado que cuenta las fichas de color blancas y negras y las retorna en CantBlancas y en CantNegras respectivamente. Los espacios vacíos encerrados por fichas de un solo color se consideran como fichas de ese mismo color.
- **cascaraEncerrarVacio(Board, Fila, Col, Color, Board):** Este predicado complementa al predicado contarFichas/3 colocando fichas de un determinado color en zonas vacías encerradas por ese mismo color Color.

## Implementación en JavaScript

En el tablero del videojuego cada celda tiene mapeado un oyente que ejecuta el predicado goMove de Prolog, en consecuencia de este predicado cae en el handleSuccess de JavaScript y actúa acorde a esta llamada, tales respuestas pueden ser: colocación de fichas, eliminación de encerradas, invalidación de jugadas.

El botón “Pasar”, luego de presionado dos veces seguidas, llama al predicado “contarFichas”, en su retorno dentro del handleSuccess se muestra los resultados de la partida y acto seguido se genera una nueva partida. Este predicado puede ser llamado por el botón “Como vamos?” para visualizar el avance de la partida.

## Observaciones generales

- Se decidió usar el Operador de corte (cut) para eliminar cálculos incensarios luego de fallas en el predicado encerrado reduciendo así drásticamente el tiempo de ejecución y eliminando pruebas de metas que van a fallar.
- En la implementación de JavaScript, dentro del método handleSuccess se decidió usar una variable global para detectar quien hizo llamadas a Prolog y actuar acorde a dicha llamada.
- En caso de empate, se tomó la decisión de que el jugador con fichas blancas será el ganador de la partida.