

Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

Centro Académico de Limón



Proyecto Programado 03 - Laberinto dirigido

Lenguajes de programación

Hansol Antay Rostrán

II Semestre, Año 2022

Fecha de entrega: 29 de octubre, 2022

# Tabla de contenidos

<b>1</b>	<b>Introducción</b>	<b>3</b>
1.1	Breve descripción del proyecto . . . . .	3
1.2	Descripción técnica del proyecto . . . . .	3
<b>2</b>	<b>Diseño del programa</b>	<b>4</b>
2.1	Directorios . . . . .	4
2.2	Algoritmos utilizados . . . . .	5
2.2.1	Búsqueda de la meta por fuerza bruta . . . . .	5
2.2.2	Determinando el camino de solución . . . . .	6
<b>3</b>	<b>Manual de usuario</b>	<b>7</b>
3.1	Requerimientos no funcionales . . . . .	7
3.2	Compilación del programa . . . . .	8
3.3	Instrucciones de uso . . . . .	8
3.3.1	Menú principal . . . . .	8
3.3.2	Ventana configuración (previo al juego) . . . . .	9
3.3.3	Ventana de juego . . . . .	10
<b>4</b>	<b>Dependencias externas</b>	<b>15</b>
<b>5</b>	<b>Análisis de Resultados</b>	<b>15</b>
<b>6</b>	<b>Fuentes digitales</b>	<b>15</b>
<b>7</b>	<b>Librerías utilizadas</b>	<b>16</b>
<b>8</b>	<b>Conclusiones</b>	<b>16</b>

## Tabla de figuras

1	Directorios del proyecto . . . . .	4
2	Búsqueda del final . . . . .	6
3	Determinando el camino de solución . . . . .	7
4	Ventana principal . . . . .	8
5	Ventana de configuración de juego . . . . .	9
6	Ventana de juego . . . . .	10
7	Verificar posición con solución . . . . .	11
8	Verificar posición sin solución . . . . .	11
9	Ventana principal . . . . .	12
10	Mostrar solución . . . . .	12
11	Ventana de estadísticas (partidas) . . . . .	13
12	Ver repetición de partida . . . . .	14
13	Lista de cotejo (según requerimientos) . . . . .	15

# 1 Introducción

## 1.1 Breve descripción del proyecto

El proyecto se basa en la resolución de laberinto dirigidos mediante una interfaz gráfica con la posibilidad de que el usuario pueda navegar entre él libremente con opciones "inteligentes" para pistas, resolución del laberinto o verificar si la posición del jugador lo puede llevar al final del laberinto. Además, se cuenta con persistencia de datos de las partidas por si el usuario desea ver repeticiones de juegos anteriores.

## 1.2 Descripción técnica del proyecto

El sistema se divide en dos partes fundamentales: parte lógica, desarrollada en Prolog para la resolución de laberintos y demás funcionalidades que ofrece el programa al momento de jugar y la interfaz gráfica, que se desarrolló utilizando Python (Tkinter) para mostrar el tablero, mostrar movimiento de las piezas, estadísticas y repeticiones de las partidas. La conexión entre estos dos lenguajes se estableció mediante la librería swiplserver.

## 2 Diseño del programa

### 2.1 Directorios

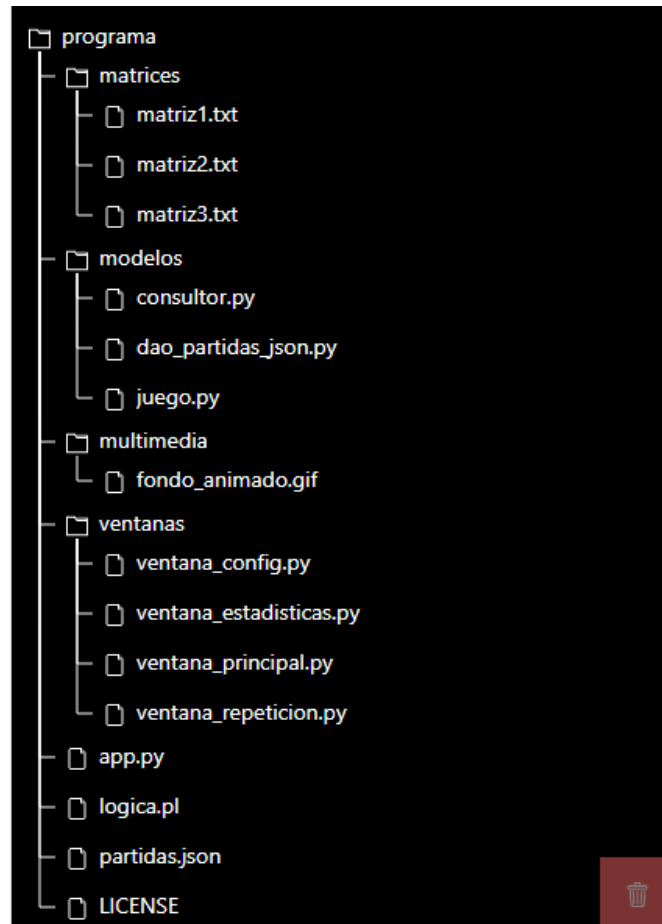


Figure 1: Directorios del proyecto

El folder **matrices** contiene archivos .txt que corresponderán a las matrices que son cargadas desde Prolog y posteriormente mostrados en la interfaz gráfica.

El folder **modelos** contiene archivos que ofrecerán interacción con el usuario, de forma más específica:

1. **consultor.py**: archivo que manejará la conexión y comunicación entre la interfaz gráfica con python y la base de conocimiento implementada en Prolog.

2. **dao\_partidas\_json.py**: este archivo manejará persistencia de datos de las partidas que se jugarán, en este irá la información de cada partida sea: el nombre del usuario, tiempo transcurrido, movimientos realizados, cantidad de sugerencias utilizadas por el usuario y si llegó a la solución, abandonó o si prefirió ver el camino para llegar a la meta.
3. **juego.py**: esta clase será el pilar fundamental para la gestión del juego. En este se define y dibuja la matriz, se detecta las teclas presionadas por el usuario para el movimiento dentro de la matriz y se manda la información al dao para que se almacenen los datos de las partidas.

El folder **multimedia** contiene el material audiovisual del proyecto, imágenes, gifs y demás que se podrá extender según se desee más detalle en el apartado audiovisual del proyecto.

El folder **ventanas** contendrán las ventanas principales para mostrar al usuario.

1. **ventana\_principal.py**: ventana de arranque de la aplicación.
2. **ventana\_config.py**: ventana de configuración previo al

## 2.2 Algoritmos utilizados

Algunos de los algoritmos utilizados en el programa se detallarán mediante gráficos y/o pseudocódigo:

### 2.2.1 Búsqueda de la meta por fuerza bruta

Primero se realizan todos los movimientos que son posibles en el laberinto y estos son definidos como posiciones visitadas. Desde el punto inicial se hacen todos los movimientos y se guardan los cruces hasta los que llegó o hasta haber llegado al final (en caso de que la meta esté pegada al inicio).

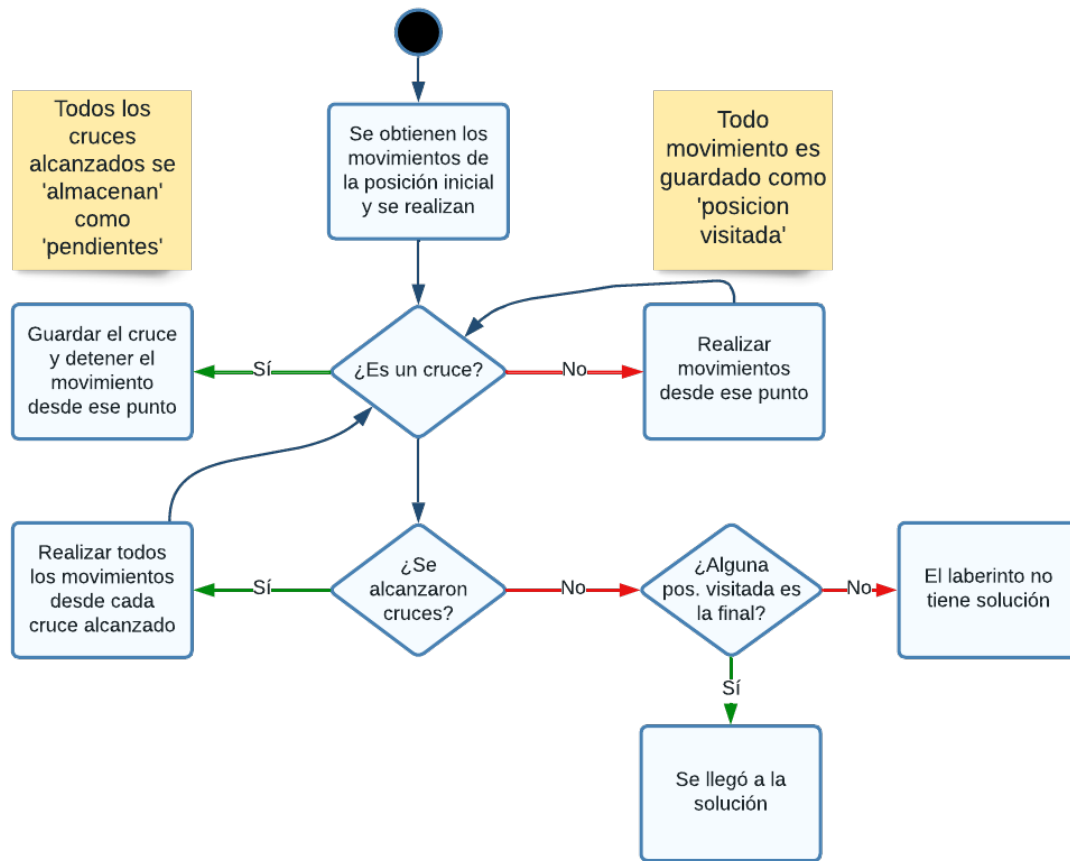


Figure 2: Búsqueda del final

### 2.2.2 Determinando el camino de solución

El problema del algoritmo anterior es que va realizando los movimientos, uno a la vez, y se van almacenando en una lista. El detalle es que dichos movimientos no pueden ser continuos, por ejemplo, puede haber llegado hasta algún punto (x, y) y no encontrar ningún otro movimiento válido para moverse, en dicho caso es una posición sin retorno y por se seguirán haciendo movimientos desde otro punto del laberinto y todo esto haciéndose de forma lineal. Por ende, la solución que se eligió fue ir recorriendo cada punto de la matriz y eliminar los puntos sin salida hasta encontrar el camino correcto que dio para la solución final del laberinto.

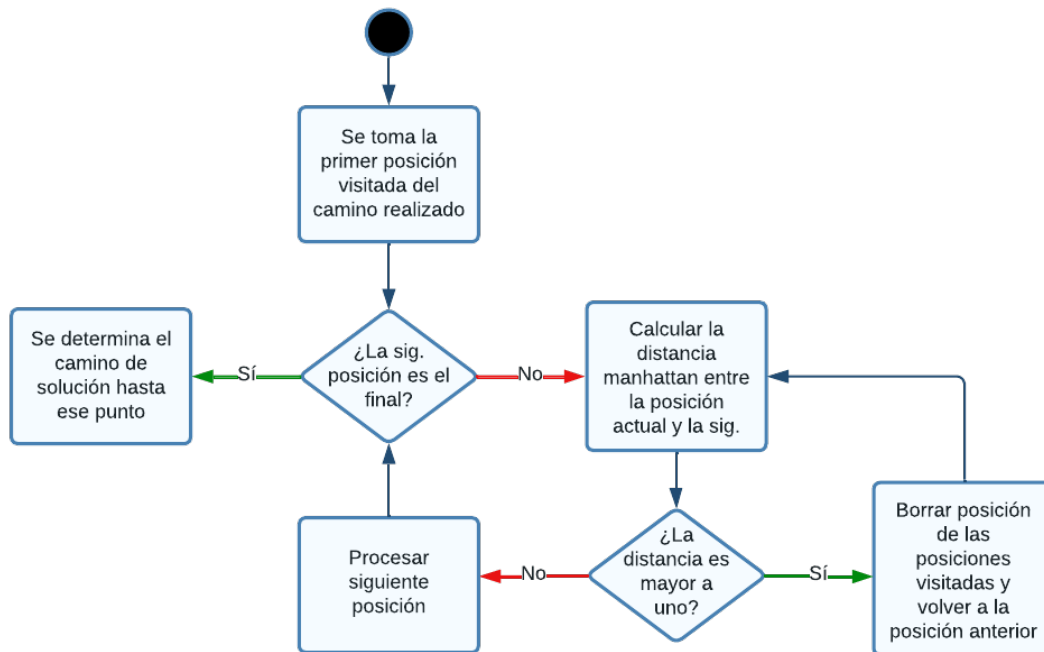


Figure 3: Determinando el camino de solución

Y todas esas posiciones que son borradas, se determinan justamente como *posiciones borradas* para que luego se muestren en la matriz como limitantes del laberinto.

## 3 Manual de usuario

### 3.1 Requerimientos no funcionales

A continuación se enumera el software y librerías que son necesarios para la ejecución del programa.

1. **Sistema operativo:** Windows 8 o superior, las pruebas se realizaron en este entorno, no queda comprobado el correcto funcionamiento en otro sistema operativo.
2. **Python:** versión 3.8.x o superior. Se puede descargar desde [la página oficial](#).
3. **SWI-Prolog:** versión 8.4.3.x utilizada para las pruebas, se puede descargar desde [la página oficial](#).



4. **Librería simplejson (Python):** versión 3.17.6 utilizada para las pruebas, se puede utilizar el comando **pip install simplejson**, más información en la [documentación oficial](#).
5. **SWIPLserver (Python):** se puede utilizar el comando **pip install swiplserver**, o seguir la [documentación oficial](#).

## 3.2 Compilación del programa

Simplemente se ejecuta el fichero **run.bat** que se encuentra junto con la carpeta programa y documentación.

## 3.3 Instrucciones de uso

### 3.3.1 Menú principal

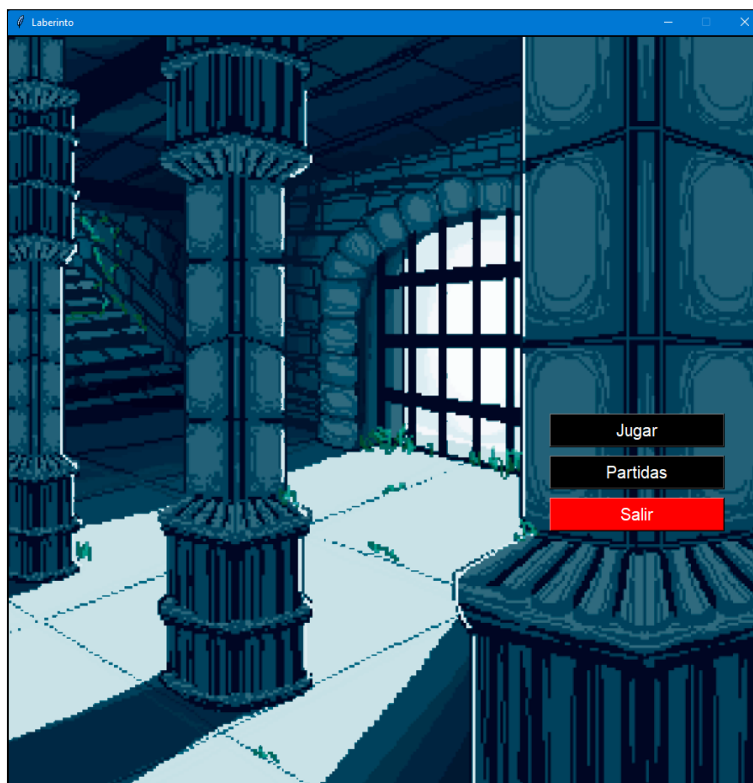


Figure 4: Ventana principal

En este menú se tienen las siguientes opciones:

1. **Jugar:** esta opción redirige a la ventanación de configuración (previo al juego) para empezar a jugar en el laberinto.
2. **Partidas:** en está opción se podrán ver las estadísticas de las partidas, además se poder visualizar las repeticiones de cada una de estas.
3. **Salir:** sale de la aplicación.

### 3.3.2 Ventana configuración (previo al juego)

En esta ventana el se debe digitar el nombre de usuario que será registrado y elegir una de las matrices que se encuentra en el folder **matrices**.



Figure 5: Ventana de configuración de juego

Y el usuario puede jugar o volver al menú principal.

### 3.3.3 Ventana de juego

En esta se muestra la matriz y el usuario tiene varias opciones a disposición que son opcionales, puede jugar utilizando las teclas (arriba, abajo, derecha izquierda) del teclado para moverse según sea válido en el laberinto.

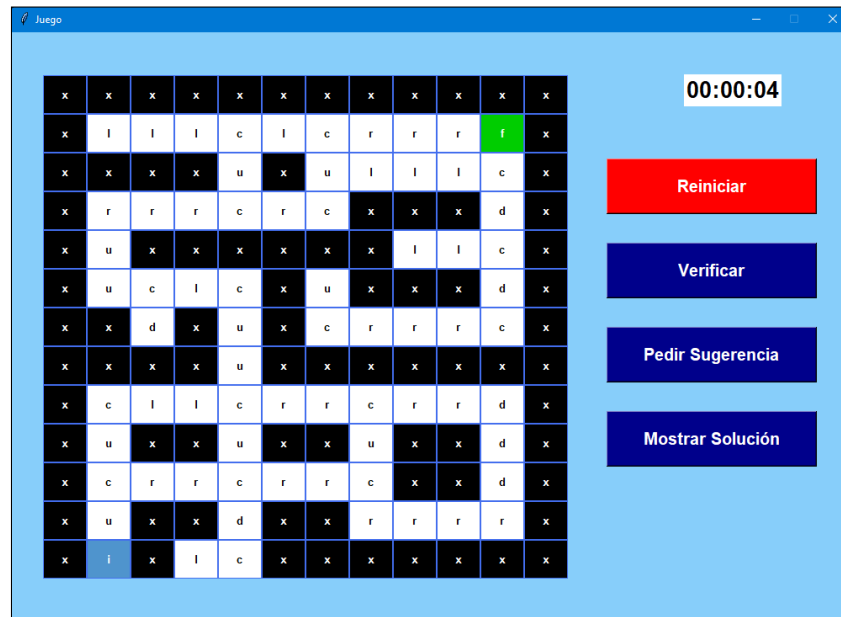


Figure 6: Ventana de juego

El usuario puede **reiniciar** el juego, simplemente se reinician los valores a su estado inicial jugando con la misma matriz y nombre de usuario elegidos previamente.

En todo momento el usuario podrá verificar si la posición en la que se encuentra lo llevará a la posición final.

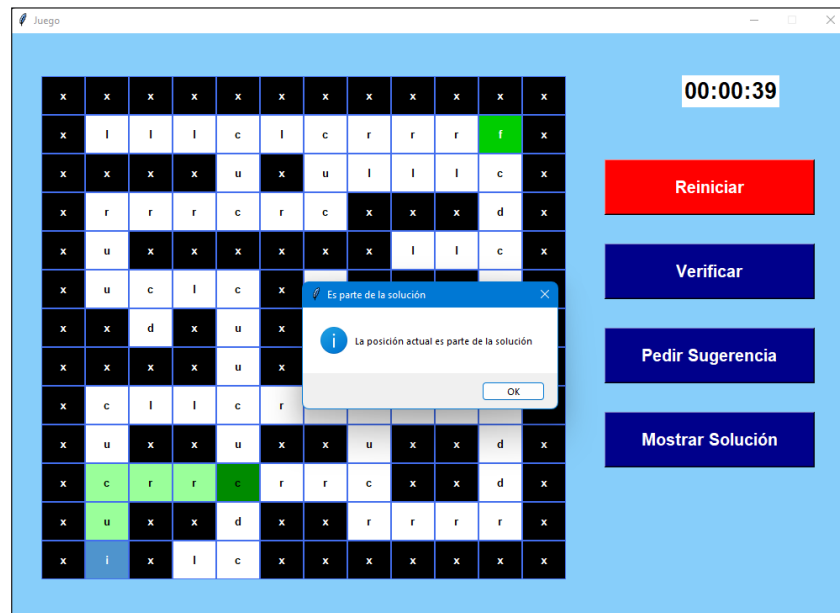


Figure 7: Verificar posición con solución

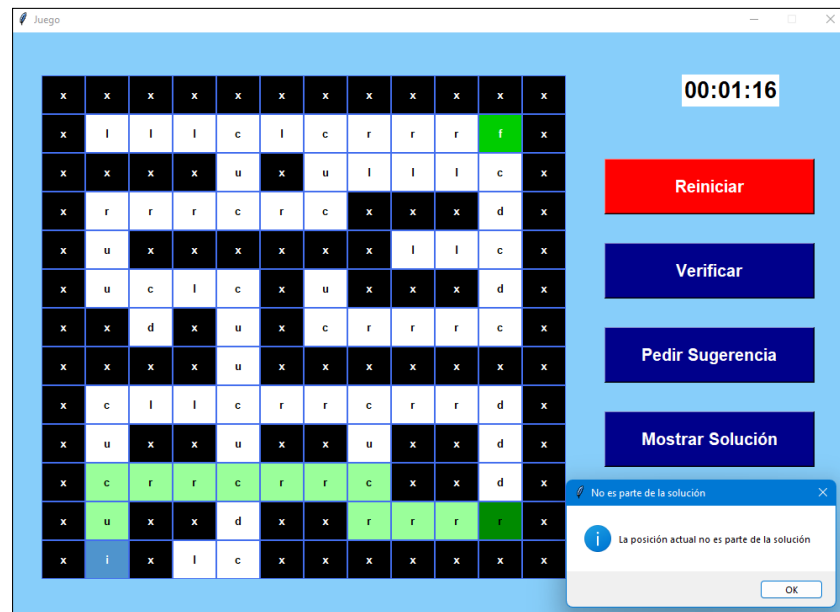


Figure 8: Verificar posición sin solución

Podrá pedir una sugerencia para realizar el movimiento, en este caso, se le pin-  
tarán las casillas las cuáles lo pueden llevar a la solución.



Figure 9: Ventana principal

Puede mostrar la solución (camino dorado) y las limitantes del laberinto (rojo).

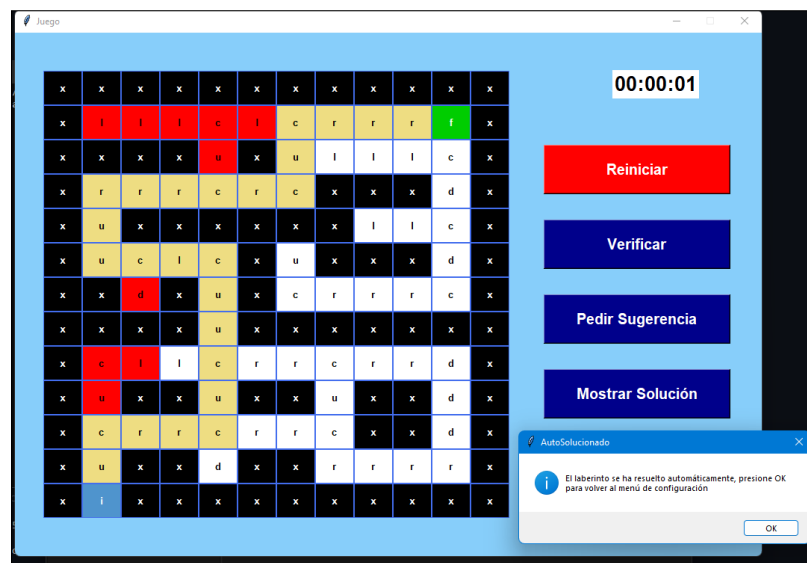
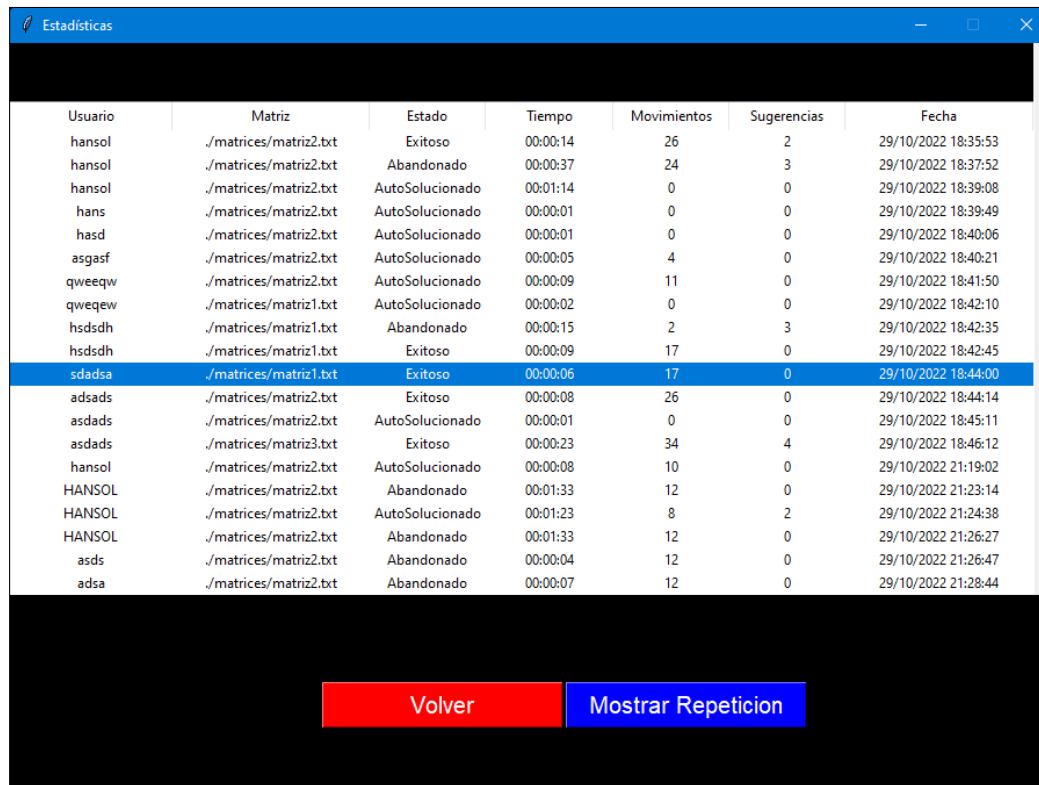


Figure 10: Mostrar solución

Regresando al **menú principal**, la opción de las partidas mostrará las estadísticas de todas las partidas jugadas recientemente con detalle, adicionalmente, el usuario podrá elegir una de las partidas y podrá ver la repetición de los movimientos realizados de dicha partida.



Usuario	Matriz	Estado	Tiempo	Movimientos	Sugerencias	Fecha
hansol	./matrices/matriz2.txt	Exitoso	00:00:14	26	2	29/10/2022 18:35:53
hansol	./matrices/matriz2.txt	Abandonado	00:00:37	24	3	29/10/2022 18:37:52
hansol	./matrices/matriz2.txt	AutoSolucionado	00:01:14	0	0	29/10/2022 18:39:08
hans	./matrices/matriz2.txt	AutoSolucionado	00:00:01	0	0	29/10/2022 18:39:49
hasd	./matrices/matriz2.txt	AutoSolucionado	00:00:01	0	0	29/10/2022 18:40:06
asgasf	./matrices/matriz2.txt	AutoSolucionado	00:00:05	4	0	29/10/2022 18:40:21
qweeqw	./matrices/matriz2.txt	AutoSolucionado	00:00:09	11	0	29/10/2022 18:41:50
qweqew	./matrices/matriz1.txt	AutoSolucionado	00:00:02	0	0	29/10/2022 18:42:10
hsdsdh	./matrices/matriz1.txt	Abandonado	00:00:15	2	3	29/10/2022 18:42:35
hsdsdh	./matrices/matriz1.txt	Exitoso	00:00:09	17	0	29/10/2022 18:42:45
sdadsa	./matrices/matriz1.txt	Exitoso	00:00:06	17	0	29/10/2022 18:44:00
adsads	./matrices/matriz2.txt	Exitoso	00:00:08	26	0	29/10/2022 18:44:14
asdads	./matrices/matriz2.txt	AutoSolucionado	00:00:01	0	0	29/10/2022 18:45:11
asdads	./matrices/matriz3.txt	Exitoso	00:00:23	34	4	29/10/2022 18:46:12
hansol	./matrices/matriz2.txt	AutoSolucionado	00:00:08	10	0	29/10/2022 21:19:02
HANSOL	./matrices/matriz2.txt	Abandonado	00:01:33	12	0	29/10/2022 21:23:14
HANSOL	./matrices/matriz2.txt	AutoSolucionado	00:01:23	8	2	29/10/2022 21:24:38
HANSOL	./matrices/matriz2.txt	Abandonado	00:01:33	12	0	29/10/2022 21:26:27
asds	./matrices/matriz2.txt	Abandonado	00:00:04	12	0	29/10/2022 21:26:47
adsa	./matrices/matriz2.txt	Abandonado	00:00:07	12	0	29/10/2022 21:28:44

VolverMostrar Repetición

Figure 11: Ventana de estadísticas (partidas)

Al momento de ver la repetición se abrirá otra ventana con el laberinto jugado por el usuario y se mostrará los movimientos (uno a la vez de como el usuario jugó la partida).

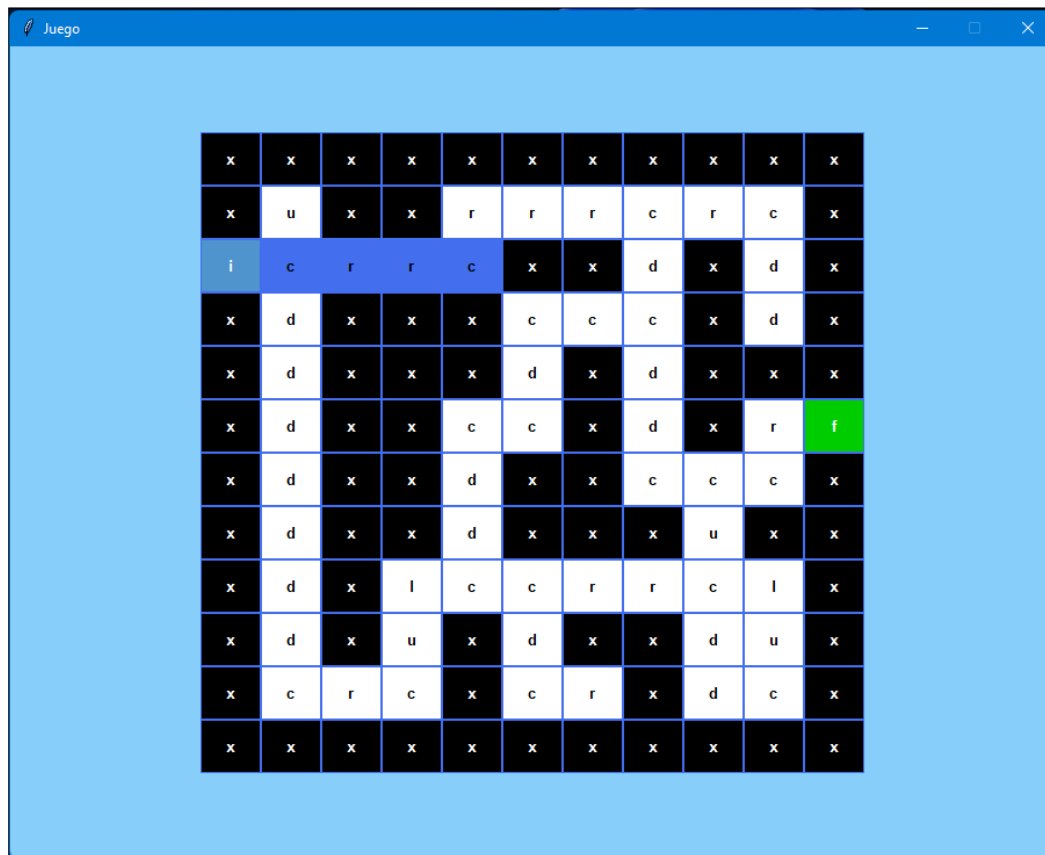


Figure 12: Ver repetición de partida

Se va pintando en azul el camino seguido por el usuario.

## 4 Dependencias externas

1. **simplejson**: librería para persistencia de datos de partidas.
2. **swiplserver**: librería para la conexión entre prolog y python.

## 5 Análisis de Resultados

A continuación se presentará una breve tabla con los lineamientos solicitados y el estado final en la entrega del producto.

L = Logrado, N = No logrado	
Requerimiento	Estado
Obtener matriz desde prolog	L
Que el usuario pueda elegir una matriz	L
Reiniciar juego	L
Verificar posición	L
Sugerir movimiento	L
Ver solución	L
Estadísticas	L
Ver repetición de partidas	L
Juegos cronometrados	L

Figure 13: Lista de cotejo (según requerimientos)

## 6 Fuentes digitales

A continuación se en listarán fuente digitales que ayudaron directa o indirectamente en el desarrollo de este programa.

1. **inductorofsoftware** [está página](#) sirvió como referencia de entrada para realizar una conexión entre la base de conocimientos y la aplicación con Python.



2. **stackoverflow** [de esta página](#) se copió el código que está como respuesta a la pregunta para la lectura de archivos línea por línea en prolog. Que sirvió directamente para recibir la matriz desde Prolog.

## 7 Librerías utilizadas

1. **Tkinter (Python)**: visualización gráfica del programa.
2. **Pio (Prolog)**: para funcionalidades de entrada y salida de ficheros.
3. **os**: para obtener directorios y rutas de archivos.
4. **simplejson**: para persistencia de datos.
5. **datetime**: para manejo de fecha y hora de partidas.

## 8 Conclusiones

Se lograron todos los objetivos solicitados con una pequeña excepción: en el momento de que se tienen más caminos de solución, el programa los encuentra y los muestra pero si el camino que junta a ambas soluciones no siempre estarán conectados.