**UNIVERSIDAD SAN PABLO DE GUATEMALA**

Facultad de Ciencias Empresariales

Escuela de Ingeniería en Ciencias y Sistemas de la Computación.

Dibujo de un perro

Descripción generada automáticamente con confianza media

Primer parcial

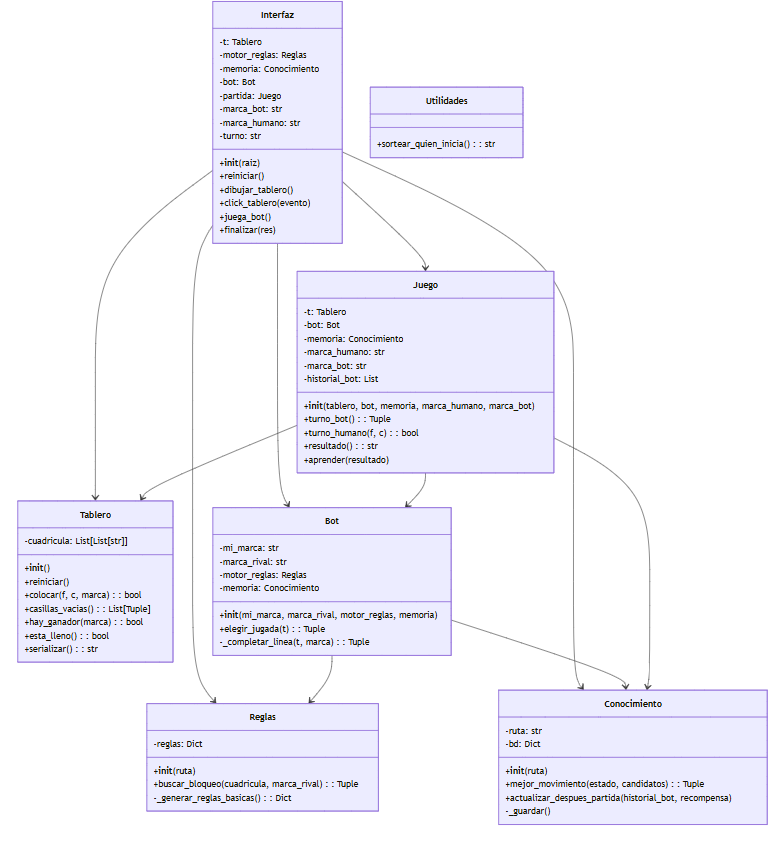
Trabajo presentado en el curso de Sistemas Expertos

Impartido por el ingeniero Marcos Alfredo Orozco

Henry Morales - 2200304

**CODIGO PYTHON JUEGO TOTITO**

**DIAGRAMA DE CLASES**



**DIAGRAMA DE COMPONENTES**

Diagrama

El contenido generado por IA puede ser incorrecto.

**EJECUCIÓN**

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.El código se compone de clases escritas en lenguaje Python, son siete clases y una carpeta “data”, esta contiene las reglas de bloque + la base de conocimiento que se va enriqueciendo con los jugadas, esta misma base es consultada para escoger la mejor jugada.

Para ejecutar el juego hágalo desde una terminal de símbolo de sistema ubicándose en la ruta donde están los archivos “py” de Python Texto

El contenido generado por IA puede ser incorrecto.

Forma

El contenido generado por IA puede ser incorrecto.*Tabla, Calendario

El contenido generado por IA puede ser incorrecto.*Para lanzar el juego ejecute el comando *python main.py* , este iniciará el tablero, aleatoriamente indicará quien incia la partida, inicie el juego y disfrute ¡

**LECTURA DE ARCHIVOS JSON**

Reglas.json: Es el “manual de defensa” del bot, contiene 24 patrones de jugadas “peligrosas” del jugador humano que deben ser bloqueadas para evitar que el bot pierda. El código lo usa durante:

1. Al iniciar: el juego carga las reglas desde el archivo
2. Durante el juego: el bot consulta sí las partidas del humano coincide con alguna de las jugadas peligrosas, si encuentra alguna jugada, determina la casilla de bloqueo.

Ejemplo: Si el humano tiene las casillas (0,0) y (0,1), el bot debe bloquear (0,2) para evitar perder.

{

"ocupadas": [[0,0], [0,1]],

"bloqueo": [0,2],

"tipo": "horizontal"

}

Base\_conocimiento.json: Es la “memoria” de la experiencia acumulada del bot, registra las partidas anteriores que lo llevaron a ganar o perder un juego. Se usa:

1. Al iniciar: se carga la base de experiencia previa.
2. Al decidir: el bot busca en su memoria; que hizo la última vez que estuvo en esa situación.
3. Después de jugar: actualiza los “puntajes” de acuerdo a sí el juego lo gano , perdió o empató.

Ejemplo: en esta situación, jugar (1,1) ha dado mejores resultados que (0,2)

"o\_\_\_x\_\_\_\_": {

"puntajes": {

"0,2": 0,

"1,1": 2

},

"recomendado": "1,1"

}

Flujo de aprendizaje:

1. El bot juega una partida
2. Si gana: aumenta los puntajes de todas sus jugadas (+1)
3. Si pierde: Disminuye puntajes de sus jugadas (-1)
4. Si empata: no cambia los puntajes.
5. Guarda la nueva experiencia en el archivo.